Contribution à l'optimisation en conception préliminaire de produit

Issam Mazhoud

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Génie Industriel**

Arrêté ministériel : 7 août 2006

Présentée par

Issam MAZHOUD

Thèse dirigée par **Jean BIGEON**
codirigée par **Khaled HADJ-HAMOU** et **Patrice JOYEUX**

préparée au sein du **Laboratoire G-SCOP**
dans **l'École Doctorale I-MEP2**

Contribution à l'optimisation en conception préliminaire de produit

Soutenance prévue le **07 Mai 2014**,
Sous réserve des rapports du jury composé de :

M. Pascal BROCHET
Professeur à l'Université de Technologie de Belfort-Montbéliard, Examinateur
M. Bernard GRABOT
Professeur à l'Ecole Nationale d'Ingénieurs de Tarbes, Rapporteur
M. Patrick SEBASTIAN
Maître de Conférences HDR à l'Université Bordeaux I, Rapporteur
M. Patrick SIARRY
Professeur à l'Université Paris-Est Créteil, Examinateur
M. Patrice JOYEUX
Docteur-Ingénieur à Hager, Co-encadrant
M. Khaled HADJ-HAMOU
Maître de Conférences à Grenoble INP, Co-encadrant
M. Jean BIGEON
Directeur de Recherche au CNRS, Directeur

Université Joseph Fourier / Université Pierre Mendès France /
Université Stendhal / Université de Savoie / Grenoble INP

Résumé

L’optimisation en conception de produit constitue une activité à forte valeur ajoutée en entreprise. Ceci est d’autant plus important qu’elle est appliquée dans les premières phases du processus de conception. Les travaux dans cette thèse se placent dans ce contexte et proposent des outils adaptés d’aide à la décision en pré-dimensionnement de produits suivant deux critères: présence ou non de fonctionnelles dans le modèle, prise en compte ou non des incertitudes dans le modèle.

Une méthode à base de calcul d’intervalles et de propagation de contraintes qui permet de faire de l’optimisation déterministe est introduite. Cette méthode permet de traiter les modèles d’optimisation sans fonctionnelles et sans prise en compte d’incertitudes. Une reformulation qui permet d’améliorer la convergence de l’algorithme est proposée. Une méthode d’optimisation stochastique à base d’essaims particulaires (PSO) est présentée pour traiter les modèles de plus grande dimension. Un nouveau mécanisme de gestion de contraintes est proposé. Cet algorithme a aussi été étendu pour traiter les problèmes de conception en présence de contraintes du type équations différentielles. Afin de traiter les incertitudes dans les modèles, une méthode d’optimisation robuste est présentée. Elle combine un algorithme d’optimisation stochastique avec une méthode de propagation d’incertitude (PoV). Cette méthode de propagation d’incertitude est étendue aux modèles incluant des fonctionnelles.

Mots-Clefs

Optimisation, Dimensionnement, Robustesse, Incertitudes

Abstract

The optimization in product design is a high added-value activity. This is all the more important when it is performed at the early stages of the design process. The work presented in this thesis is placed in this context. It proposes adapted decision making tools in preliminary design following two criteria: whether or not the model contains functionals, and whether it takes into considerations the uncertainties.

A method based on interval arithmetic and constraint propagation allowing to perform deterministic global optimization is introduced. This method allows handling optimization models without functionals and without considering uncertainties. A reformulation that permits to improve the algorithm convergence is proposed. A stochastic optimization method based on particular swarms is introduced in order to handle higher dimensional problems. A new constraint handling mechanism is proposed and tested on engineering problems. This algorithm has also been extended to design problems with ordinary differential equations constraints. In order to consider uncertainties, a robust optimization method is introduced. It combines a stochastic optimization method with an uncertainty propagation method called PoV. An extension of PoV to models involving functionals is introduced.

Key Words

Optimization, Preliminary design, Robustness, Uncertainties

Table des matières

Liste des figures

Liste des tableaux

1. Introduction

1.1. Contexte & Problématique

Cette thèse porte sur l'optimisation en pré-dimensionnement de produits. Ce travail se déroule en étroite collaboration avec l'entreprise Hager. Hager Group compte parmi les leaders dans les systèmes d'installations électriques pour les logements et les bâtiments tertiaires. Le groupe compte 11400 collaborateurs et 22 sites de production dans 20 pays. Il a fait un chiffre d'affaire de 1.6 milliards d'euros en 2012.

Le groupe englobe plusieurs marques achetées au cours de son histoire mais la marque Hager représente son cœur de métier. Cette marque couvre une large gamme de produits et de services, de la distribution d'énergie électrique à la gestion technique des bâtiments en passant par le cheminement de câbles et les dispositifs de sécurité.

Les produits sur lesquels va porter mon étude sont les produits de protection électrique des biens et des personnes. Ces produits sont fabriqués en France sur les sites d'Obernai, Bischwiller et Saverne ou assemblés à Huizhou (chine) et à Rio (Brésil). Cette gamme de produits est composée des disjoncteurs et des produits différentiels.

Les disjoncteurs permettent de protéger l'installation électrique des surcharges et des courts circuits. Les produits différentiels sont destinés à mesurer les fuites de courants dans les installations électrique et de couper l'alimentation une fois le courant de fuite atteint une certaine limite. En fonction de cette limite, l'interrupteur différentiel protège les personnes (<30 mA) et/ou les biens (>30 mA).

Dans l'activité de conception, l'ingénieur part toujours d'un ensemble d'exigences et de besoins plus ou moins définis. Afin d'aboutir à une solution viable, plusieurs choix doivent être fait dans un environnement physique et industriel contraint et incertain. Ceci s'exprime par la nécessité d'optimiser un ou plusieurs critères (économiques, performances…) tout en s'assurant du respect de contraintes physiques, industrielles, normatives, de cahier des charges etc…

Plusieurs auteurs ont travaillé sur la théorie de la conception en proposant plusieurs descriptions. Pahl et al proposent une approche systématique de la conception (voir Figure 1-1) [Pahl G. et al., 2007]. Dans cette approche, l'ingénieur R&D part des besoins du client afin d'aboutir à un produit qui satisfait au mieux ces exigences.

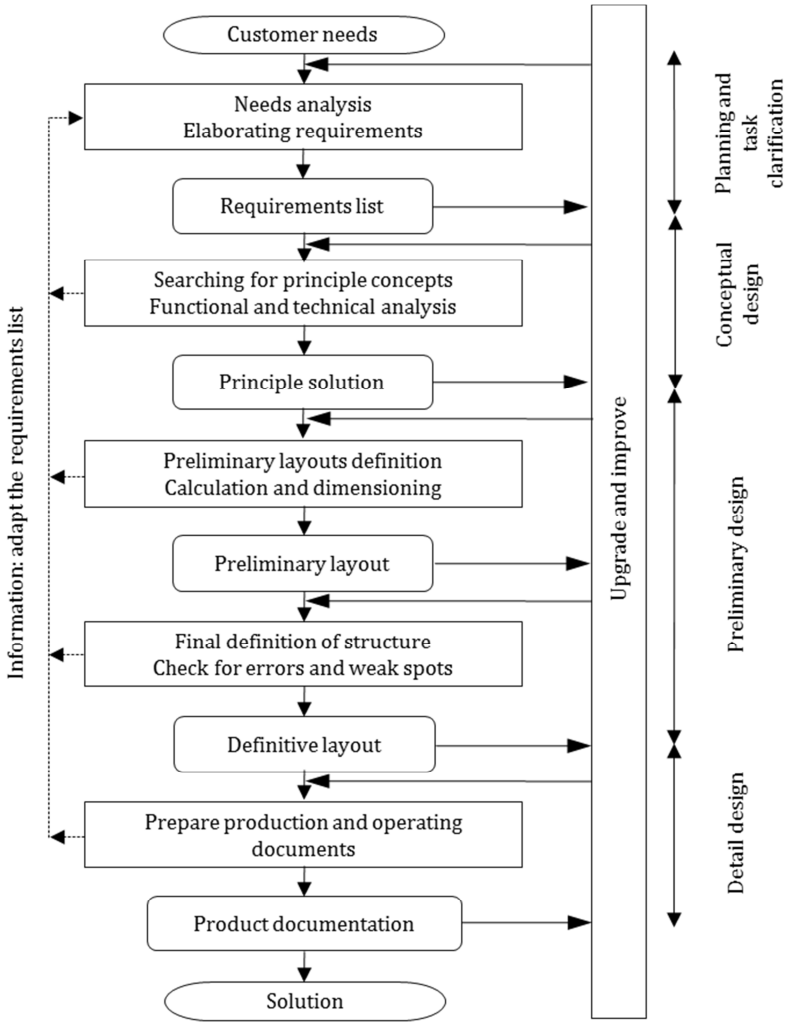

Figure 1-1. Etapes de la conception [Pahl G. et al., 2007]

Cette approche peut être divisée en 4 grandes étapes :

- Clarification des exigences client : exprimer en de termes plus clairs, plus précis les besoins du client.
- Recherche de concepts : établissement d'une liste de concepts susceptibles de répondre aux besoins.
- Conception préliminaire (ou pré-dimensionnement) : choix des concepts, d'une structure, et définition des propriétés des composants.
- Conception détaillée : étude plus détaillée de la structure du produit (éléments finis…), prototypage et tests de validation, industrialisation.

Dans la réalité, ces étapes ne sont pas forcément respectées pour plusieurs raisons plus ou moins logiques. Dans le contexte industriel, il est possible de faire un dimensionnement avec des modèles analytiques/numériques ou faire un développement par prototypage. Ce dernier cas est généralement utilisé pour adapter un produit et/ou une fonction déjà existants à un nouveau produit. Cela demande évidemment une démarche par essai/erreur. Cette démarche

se justifie lorsqu'il y'a un manque d'information, de modèles suffisamment précis sur les phénomènes mis en jeux. Dans la réalité, il peut arriver que certains phénomènes ne soient pas modélisables ce qui rend la démarche de dimensionnement par les essais nécessaire. Cette pratique réduit considérablement la marge de manœuvre du concepteur très tôt dans les étapes de développement. En effet, cette solution suppose souvent un choix de concepts et d'une géométrie bien précis. De plus, les temps de calculs sont souvent considérables ce qui rend l'approche par essai/erreur peu pratique. Toutefois, lorsqu'il est possible d'écrire un modèle suffisamment fidèle à la réalité, il est toujours plus intéressant d'explorer des solutions théoriques avant de passer au prototypage.

Les travaux de cette thèse se situent dans l'étape de pré-dimensionnement. En conception préliminaire, l'ingénieur utilise des modèles pour simuler le comportement du produit et prévoir ses performances. Les modèles de conception en pré-dimensionnement servent de base pour l'évaluation des performances d'une configuration déterminée. Ce sont des modèles paramétrés souvent multi-physiques qui simulent les performances du produit. Ils peuvent être d'un niveau de détail et de complexité plus ou moins important en fonction du besoin et des ressources disponibles. Ce modèle sera d'autant plus précis et donc proche de la réalité qu'il prend en compte des paramètres. Ici, c'est un compromis qu'il faudra faire entre la précision souhaité et la rapidité de calcul, importants en optimisation. En effet, l'optimisation sera d'autant plus rapide et efficace que le modèle est rapide à évaluer. Un modèle de conception en pré-dimensionnement évolue souvent en même temps que la définition de la solution finale devient plus précise. Ceci s'explique par le besoin de valider au mieux la solution avant de passer à l'étape de conception détaillée et prototypage.

En plus du modèle de comportement physique, l'optimisation doit prendre en compte le cahier des charges. Le cahier des charges permet de définir les contraintes à respecter ainsi que le/les objectifs à optimiser. La Figure 1-2 explique la démarche de dimensionnement. Dans la phase de reformulation, le modèle multi-physique et le cahier des charges sont combinés pour former à un modèle de dimensionnement. L'algorithme d'optimisation traite ensuite ce modèle de dimensionnement pour générer la solution optimale.

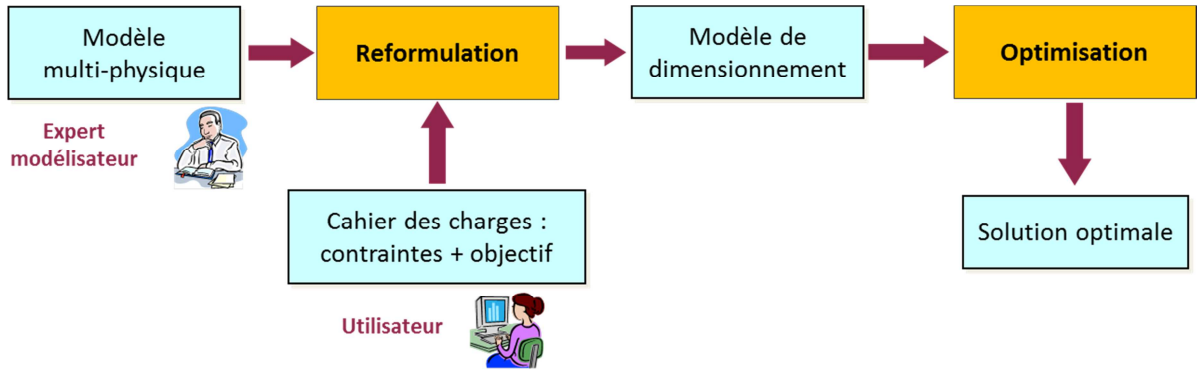

Figure 1-2. Démarche de dimensionnement

Il est important de noter la différence qui existe entre une démarche de simulation et une démarche de conception. En effet, la simulation consiste à donner des valeurs en entrée d'un modèle de simulation paramétré et à récupérer les sorties qui correspondent aux performances du produit. La conception est à l'opposé de ça. Elle consiste à fixer des exigences au niveau de paramètres de performances (sorties) et de chercher les bons paramètres de conception (entrées). Dans les deux cas, le modèle a le même objectif, c'est juste l'angle de vue qui change.

Dans la théorie, il est possible d'utiliser la démarche de simulation pour la conception en passant par une méthode par essais/erreurs. Toutefois, cette approche est souvent lente et fastidieuse spécialement lorsque le modèle de simulation contient des fonctionnelles (équations différentielles, intégrales …). Par exemple dans le modèle de l'actionneur différentiel introduit dans le paragraphe 1.2.2, une équation différentielle permet de calculer la trajectoire du noyau mobile. Tout changement des paramètres de conception impacte le comportement de ce noyau ce qui rend la démarche par essais/erreurs particulièrement difficile.

Contrairement à la méthode de modélisation par éléments finis, les modèles analytiques permettent une plus grande flexibilité et des temps de calculs plus faibles au détriment de la précision. Malgré ce manque de précision qui est, soit dit au passage, relatif, ces modèles répondent parfaitement à la complexité de l'étape de pré-dimensionnement et permettent de sélectionner les concepts et la structure du produit final. En effet, les modèles analytiques permettent la prise en compte de l'aspect multi-physique très facilement sans passer forcément par une combinaison de plusieurs logiciels de calcul. Les modèles analytiques sont également plus rapides à calculer ce qui constitue un critère primordial lors de l'optimisation.

Dans l'étape de pré-dimensionnement, le concepteur fait le choix des concepts et de l'architecture qui va permettre au produit de remplir les fonctions exigées. Il va jusqu'au choix des caractéristiques des composants, des propriétés des matériaux et des dimensions géométriques en se basant sur un ou plusieurs modèles plus ou moins précis. En effet, l'ingénieur part d'un modèle grossier qui sera affiné au fur et à mesure de la définition du produit. Cette étape est particulièrement importante car elle permet de choisir les concepts qui seront utilisés et définit les principaux paramètres du produit final. Selon [Berliner C. et al., 1988] et [Scaravetti D., 2004], la conception préliminaire conditionne 70% à 80% du coût total du cycle de vie du produit (voir Figure 1-4 et Figure 1-3). Par conséquent, l'optimisation dans cette étape représente un enjeu très important.

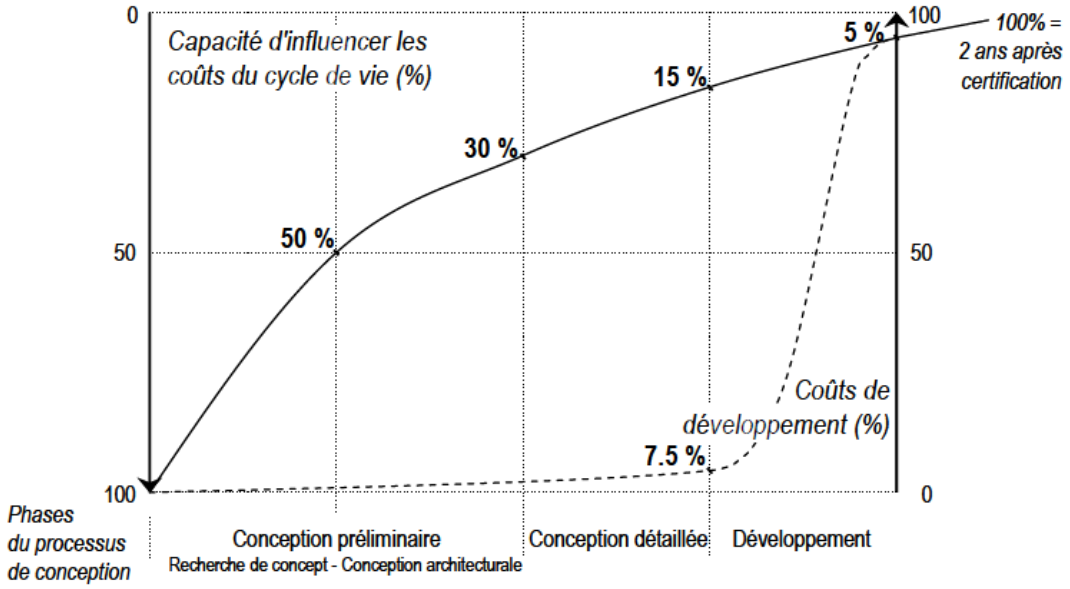

Figure 1-3.Coûts de développement vs Coûts engagés dans le processus de conception [Scaravetti D., 2004]

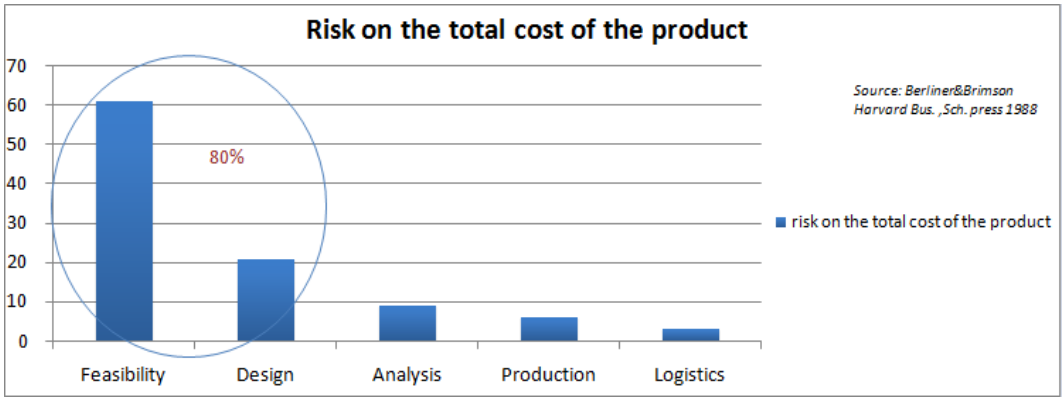

Figure 1-4. Coûts engagés pour les étapes de la conception [Berliner C. et al., 1988]

D'un autre côté, les ingénieurs en développement de produit ont un besoin croissant en précision. C'est pour cette raison qu'ils combinent souvent plusieurs modèles, de différents domaines de la physique afin de simuler le couplage entre différents phénomènes issues de physiques ou de domaines différents.

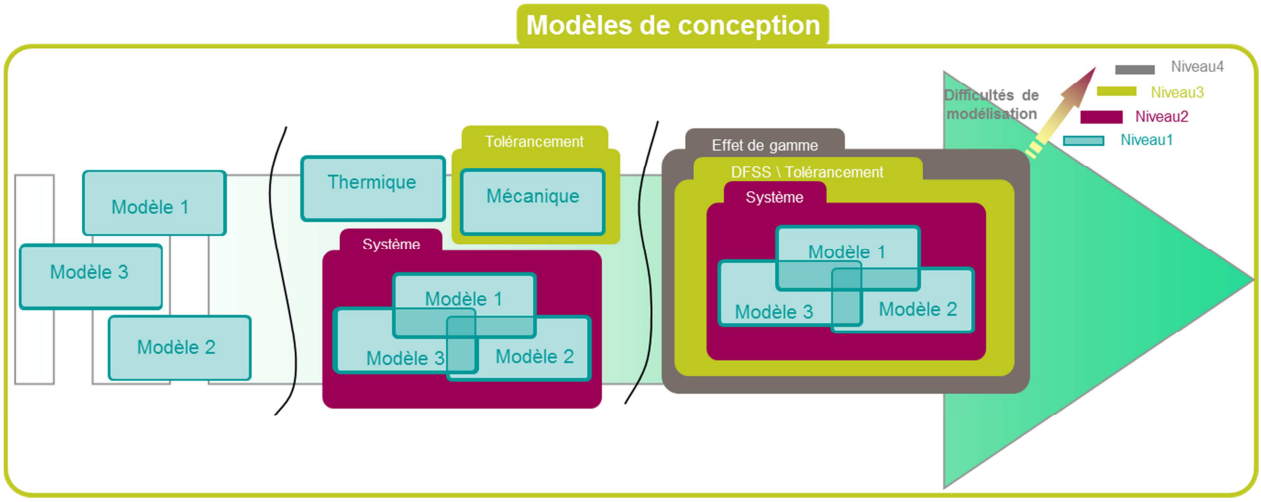

Figure 1-5. Modèles de conception : vers la vision système

Ce besoin croissant en complexification est illustré par la Figure 1-5. En effet, différents modèles sont combinés afin de s'approcher de plus en plus du comportement réel du produit, dans ses différentes conditions de fonctionnement. L'ingénieur doit donc raisonner en termes de système en combinant des modèles qui peuvent être issus de différents logiciels.

Dans ce travail, on se positionne à l'étape de pré-dimensionnement. On considère des modèles de conception analytiques :

- Rapides : ces modèles sont en règle générale rapides à calculer.
- Equations analytiques : les modèles sont basés sur des équations analytiques issues des lois de la physique. A la différence des modèles éléments finis, on se place au niveau des composants et du système.
- Boite blanche/noire : lors de l'optimisation, ces modèles peuvent être du type boite blanche où on accède aux équations du modèle, ou boîte noire. Ceci est conditionné par le type de modélisation adopté et les logiciels utilisés qui peuvent être fermés.

Ce choix a un impact sur les méthodes d'optimisation développées. Les méthodes qui seront développées seront également valables pour les modèles du type boite noire. Par conséquent, elles seront applicables aussi aux modèles éléments finis.

Les modèles analytiques décrivent le comportement physique du produit. Ce modèle est établi par des experts en modélisation ayant des compétences variées ou par des outils spécifiques. Il comporte un ensemble d'équations analytiques faisant intervenir plusieurs paramètres. Ces modèles ont différentes caractéristiques et niveaux de complexité :

  - Ces modèles impliquent généralement de 10 à 1000 paramètres avec au tour de 20 degrés de liberté.
  - Ils sont constitués en grande partie par des équations du type égalité (contraintes d'égalité) qui reflètent les lois de la physique. Par conséquent, ces contraintes sont relativement difficiles à relâcher

- La formulation mathématique des contraintes peut être explicite, implicite ou faisant intervenir des fonctionnelles (équations différentielles, intégrales, MinMax…)
- Domaines de variabilité : le modèle physico-mathématique peut spécifier le domaine de variation de certains paramètres.
- Le modèle est multi-physique, il peut découler de plusieurs physiques (mécanique, électrique, thermique…).
- Les contraintes sont non linéaires, ayant des formes différentes selon le domaine d'application : logarithme, exponentielle, cosinus, complexes…
- Les expressions mathématiques sont, pour la plupart, continues et dérivables.
- La convexité des équations n'est pas connue et difficilement démontrable.

En fonction de leur niveau d'abstraction, les modèles de pré-dimensionnement permettent souvent d'avoir des solutions intéressantes tout en étant simples. Le modèle du moteur à aimants permanents présenté dans le paragraphe 1.2 n'a que 10 équations et permet d'avoir une idée sur les grandeurs caractéristiques du produit final.

Les modèles analytiques en pré-dimensionnement peuvent avoir plusieurs caractéristiques : linéarité, convexité, contraintes du type équations différentielles (ODE), boite noire/blanche etc… Ces caractéristiques ont un impact sur le choix de l'algorithme d'optimisation le mieux adapté. Il faut que ce choix prenne en compte la nature des contraintes, de la fonction objectif, le nombre de variables … afin de choisir le meilleur algorithme susceptible de donner les meilleures performances (temps et précision).

Les propriétés du modèle de conception ont aussi une incidence directe sur la complexité du modèle. Cette complexité s'exprime par un temps de calcul plus important. Egalement, il devient plus difficile d'évaluer l'impact de chaque paramètre sur les performances finales du produit sans l'aide d'outils adaptés.

Dans ce travail, on considère que les modèles développés par les ingénieurs de conception sont suffisamment fidèles à la réalité et ne seront donc pas remis en question. Ainsi, il peut être intéressant de tenir compte des variabilités auxquelles ces modèles sont assujettis. Ceci permet de voir l'impact des variabilités des paramètres de conception sur les performances du produit final. L'objectif étant de s'assurer que les solutions développées respectent le cahier des charges et tiennent les performances souhaitée, quelques soient les variabilités des paramètres d'entrée. En conséquence, deux classes d'optimisation se dégagent : l'optimisation classique et l'optimisation robuste.

Les modèles de conception en pré-dimensionnement peuvent donc être classés selon leurs difficultés et le besoin en optimisation. Le travail de la thèse s'articule autour des modèles de pré-dimensionnement avec différents niveaux de complexité. L'idée est de proposer à chaque fois un outil adapté aux spécificités de chaque modèle.

On a identifié deux critères pour caractériser la complexité des modèles et qui guiderons le choix des algorithmes développés :

- Présence de fonctionnelles : dans ce travail, on s'intéresse principalement aux équations différentielles. Donc on traitera des modèles avec ou sans équations différentielle.
- Prise en compte des incertitudes : ce qui amène à faire de l'optimisation robuste. On proposera des outils pour l'optimisation classique sans robustesse et pour l'optimisation robuste.

Ce qui donne au final quatre classes de modèles. Chacune des quatre classes implique un ou des outils d'optimisation adaptés. Le cheminement de ce mémoire suivra ainsi ces quatre classes. Cette thèse se place donc dans le contexte des modèles analytiques de pré-dimensionnement et vise à proposer des outils et des méthodes adaptées à différents types de modèles afin de pallier les problèmes de non-convergence en optimisation. En optimisation, on traite le cas mono-objectif. Le cas multi-objectif étant une couche au-dessus relativement indépendante de l'algorithme d'optimisation en lui-même.

Le plan du manuscrit suivra la matrice ci-dessous. Dans le chapitre 2, il sera question d'optimisation classique sans prise en compte d'incertitudes et sans fonctionnelles. Dans ce chapitre, une méthode déterministe d'optimisation globale à base de calcul d'intervalles et de propagation de contraintes qui cherche à garantir l'optimalité de la solution sera développée. Une reformulation qui permet d'accélérer le processus d'optimisation sera introduite.

Le chapitre 3 sera consacré à l'optimisation par essaims particulaires (PSO). Il sera question de modèles sans prise en compte d'incertitudes. Cette approche permettra de traiter des modèles de plus grande dimension. Deux approches seront développées, un PSO pour les modèles sans équations différentielles (ODE) et un PSO qui prend en compte les contraintes du type ODE.

Dans le chapitre 4, il sera question de robustesse. Des travaux sur l'optimisation robuste sur les modèles sans ODE seront présentés. Ensuite, une extension aux modèles avec équations différentielle sera introduite.

	Sans fonctionnelles	Avec fonctionnelles
	Chapitres 2&3	Chapitre 3
Sans incertitudes	• IBBA o Propagation de contraintes o Reformulation • CVI-PSO o Mécanisme de gestion de contraintes	• PSO-RK44 o CVI-PSO o Runge-Kutta
	Chapitre 4	Chapitre 4
Avec incertitudes	• CVI-PSO + PoV o PSO o Propagation d'incertitudes : PoV	• PSO-RK44 + PoV-RK44 o PoV sur modèles dynamiques o PSO-RK44

1.2. Modèles de conception : deux exemples

Nous présentons deux exemples de modèles de dimensionnement essentiellement du domaine du génie électrique. Ces modèles serviront d'exemples applicatifs aux différents algorithmes d'optimisation introduits dans les prochains chapitres.

1.2.1. Le moteur à aimant permanents

Comme exemple de modèle sans contraintes du type fonctionnelles, un modèle de moteur à aimants permanents est présenté. Il met en jeux des lois de l'électricité, le magnétisme, la mécanique et la thermique. La structure appelée MAPSE (Machine à Aimants Permanents Sans Encoches) est modélisée par les relations exprimant la conversion électromagnétique et la conservation du flux [Kone A.D. et al, 1993] [Messine F. et al, 1998]. La signification des paramètres de conception impliqués dans ce modèle est détaillée dans le Tableau 1-1et le Tableau 1-2.

Ainsi, le couple électromagnétique est donné par la relation suivante :

$$\Gamma_{em} = \frac{\pi}{2\lambda}\left(1 - K_f\right)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e$$

L'échauffement de la machine E_{ch} est défini par la relation suivante :

$$E_{ch} = k_r E J_{cu}^2$$

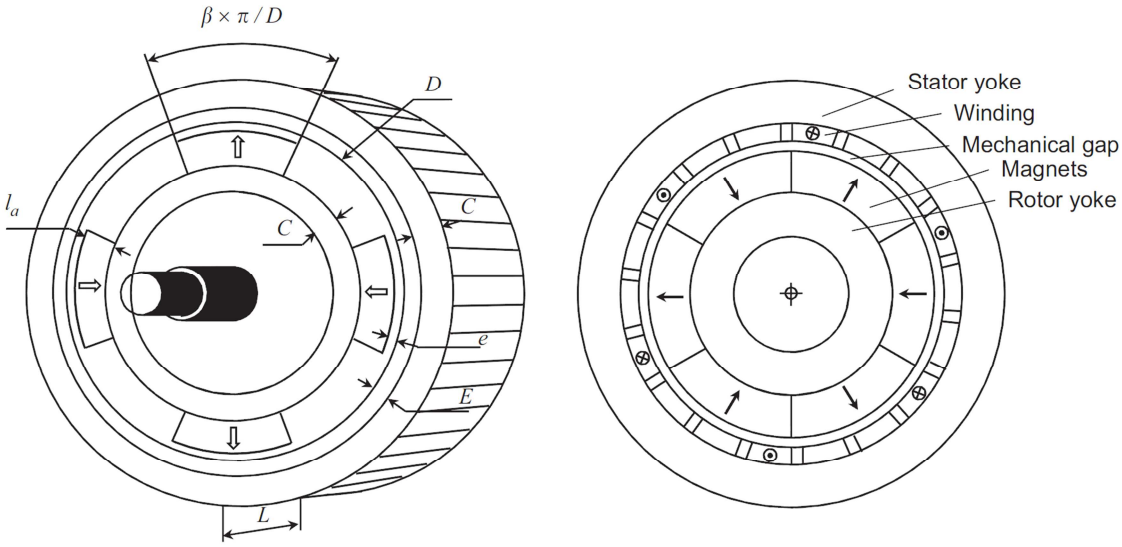

Figure 1-6. Structure MAPSE [Kone A.D. et al, 1993]

Une relation empirique reliant le coefficient de fuites magnétiques et les dimensions géométriques a été établie, avec p le nombre de paires de pôles :

$$K_f \cong 1.5 p \beta \frac{e + E}{D}$$

Le champ magnétique dans l'entrefer, supposé exclusivement radial est donné par :

$$B_e = \frac{2l_a M}{Dlog\left[\frac{D+2E}{D-2(l_a+e)}\right]}$$

On donne ici, l'expression de l'épaisseur des culasses, dans laquelle on assume que les fuites inter-polaires sont négligeables :

$$C = \frac{\pi\beta B_e}{4pB_{fer}} D$$

Le nombre de paires de pôles est donné par :

$$p = \frac{\pi D}{\Delta_p}$$

Le modèle structurel est formé uniquement de contraintes d'égalité. Les contraintes d'inégalité sont comprises dans les domaines de variation des paramètres de sortie. Ces domaines sont donnés dans le Tableau 1-1 et le Tableau 1-2.

Tableau 1-1. Paramètres variables du MAPSE

Paramètres variables		Intervalles
Diamètre d'alésage	$D(m)$	[0.001 ; 0.5]
Champ magnétique dans l'entrefer	$B_e(T)$	[0.1 ; 1]
Coefficient de fuites inter-polaires	K_f	[0.01 ; 0.3]
Densité de courant dans le cuivre	$J_{cu}(A/m^2)$	[10^5 ; 10^7]
Entrefer mécanique	$e(m)$	[0.001 ; 0.005]
Epaisseur des aimants	$l_a(m)$	[0.001 ; 0.05]
Epaisseur du bobinage	$E(m)$	[0.001 ; 0.05]
Epaisseur des culasses	$C(m)$	[0.001 ; 0.05]
Coefficient d'arc polaire	β	[0.8 ; 1]
Facteur de forme de la machine	λ	[1 ; 2.5]

Tableau 1-2. Paramètres fixes du MAPSE

Paramètres fixes		Valeurs
Coefficient de remplissage du bobinage	k_r	0.7
Induction dans le fer	$B_{fer}(T)$	1.5
Echauffement de la machine	$E_{ch}(A/m)$	10^{11}
Couple électromagnétique	$\Gamma_{em}(N.m)$	10
Aimantation des aimants	$M(T)$	0.9
Double pas polaire	$\Delta_p(m)$	0.1

L'objectif de l'optimisation est de minimiser le volume des parties utiles :

$$V_u = \pi \frac{D}{\lambda}(D + E - e - l_a)(2C + E + e + l_a)$$

Pour plus de détails sur le modèle voir [Kone A.D. et al, 1993].

Le modèle d'optimisation est l'agrégation de toutes ces équations avec les domaines de variation des paramètres de conception :

$$min\, V_u = \pi \frac{D}{\lambda}(D + E - e - l_a)(2C + E + e + l_a)$$

$$s.c \begin{cases} \Gamma_{em} = \dfrac{\pi}{2\lambda}\left(1 - K_f\right)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e \\ E_{ch} = k_r E J_{cu}^2 \\ K_f = 1.5 p \beta \dfrac{e + E}{D} \\ B_e = \dfrac{2 l_a M}{D log\left[\dfrac{D + 2E}{D - 2(l_a + e)}\right]} \\ C = \dfrac{\pi \beta B_e}{4 p B_{fer}} D \\ p = \dfrac{\pi D}{\Delta_p} \end{cases}$$

Le modèle final consiste en 6 contraintes non linéaires et 16 paramètres. Le modèle à optimiser est multi-physique. En effet, il combine des contraintes issues de différents champs de l'ingénierie : électrique, magnétique, mécanique et thermique. Les contraintes sont non linéaires, non convexes et continûment différentiables. Dans ce modèle, les paramètres sont continus et l'espace de recherche initial est donné par le Tableau 1-1et le Tableau 1-2.

1.2.2. L’actionneur différentiel

Le deuxième modèle est un modèle de pré-dimensionnement d’un actionneur différentiel avec contraintes fonctionnelles du type équations différentielles. L’actionneur consiste en une bobine et un circuit magnétique qui lui-même est constitué d’une partie mobile et d’une partie fixe comme dans la Figure 1-7. La modélisation du dispositif nécessite la prise en compte de l’aspect multi-physique incluant les aspects électrique, magnétique et mécanique [Darnault R., Bigeon J., 2005].

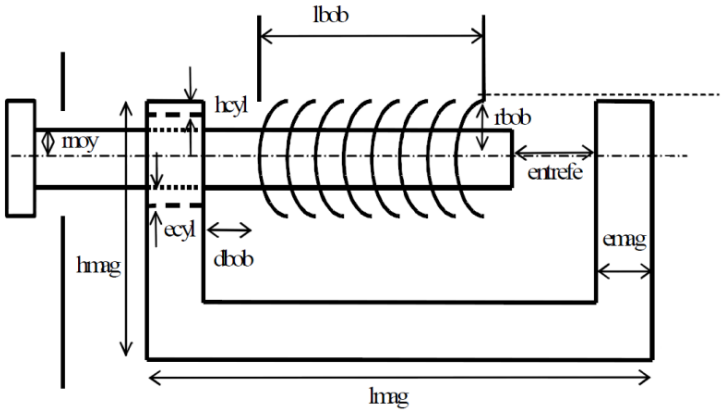

Figure 1-7. Structure de l’actionneur différentiel [Darnault R., Bigeon J., 2005]

La bobine est constituée de deux galettes et est alimentée par un circuit électrique. Le thyristor de ce circuit peut être contrôlé à tout moment. Par conséquent, il n’est pas possible de prévoir à priori la forme de signal d’alimentation. La résistance de la bobine elle-même est supposée non-négligeable. Elle est donc ajoutée au circuit électrique équivalent.

Pour des raisons pratiques, seules les équations physiques principales seront présentées ici. Les phénomènes physiques impliqués dans ce cas sont exprimés simplement par deux équations différentielles à coefficients non-constants :

$$\begin{cases} U(t) = R.i(t) + \dfrac{d}{dt}\big(L(i,z).i(t)\big) \\ m\dfrac{d^2z}{dt^2} = F(i,z) \end{cases}$$

Avec :

- i le courant dans la bobine
- z la longueur de l’entrefer
- R la résistance de la bobine
- L l’inductance de la bobine
- m la masse du noyau
- F la force globale exercée sur le noyau (magnétique + charge + frottement…)

La première équation décrit la loi de comportement électrique, la seconde la loi de comportement mécanique. La seconde équation différentielle du deuxième ordre est éclatée en deux équations différentielles ordinaires du premier ordre. En toute rigueur, il manque la loi de comportement magnétique. Cette loi est résolue lorsqu'on est capable de calculer l'inductance et les forces en fonction du courant dans la bobine et la longueur de l'entrefer. Ceci conduit à considérer que les phénomènes magnétiques sont suffisamment rapides en comparaison avec les constantes de temps mécanique et électrique.

Afin de pouvoir résoudre le système d'ODE, il est reformulé comme suit :

$$\begin{cases} \dfrac{di}{dt} = \dfrac{U(t) - Ri(t) - i(t)v(t)\dfrac{\partial L}{\partial z}}{i(t)\dfrac{\partial L}{\partial i} + L(i,z)} \\ \dfrac{dv}{dt} = \dfrac{F(i,z)}{m} \\ \dfrac{dz}{dt} = v \end{cases}$$

Un réseau de réluctances a été utilisé afin de modéliser le système. Le modèle final consiste en 37 contraintes d'égalité non-linéaires, un système d'ODE de 3-dimensionnel et 61 variables.

L'objectif de l'optimisation est de minimiser le volume en respectant ces contraintes :

- Fonctionner à n'importe quelle tension entre 50 et 400v sans se détruire,
- Fermer en moins de 30 millisecondes sans se détruire,
- Supporter la foudre sans se détruire.

Pour des raisons issues de standards de sécurité, l'actionneur doit fermer en moins de 30ms. Mais avec l'alimentation de l'actionneur qui supprime la demi-période négative (diode), le fonctionnement de l'actionneur peut ne prendre place que sur 10ms (pire cas). Par conséquent, l'actionneur doit fermer en moins de 10ms. Cette contrainte est représentée par un système d'équations différentielles qui doit être résolu pour la vérifier.

2. Optimisation globale déterministe : une nouvelle reformulation

Entre le service Marketing qui fournit un cahier des charges qui établit des contraintes à respecter, et l'ingénieur qui cherche à satisfaire ces contraintes, il peut être intéressant de proposer au concepteur un outil qui lui permet de prouver l'existence ou la non-existence de solution faisable. C'est dans ce contexte que les méthodes déterministes d'optimisation globale trouvent leur place. L'intérêt des méthodes déterministes est qu'elles proposent une garantie de l'optimum. En effet, elles proposent au concepteur un outil lui permettant de s'assurer de l'optimalité d'une solution, ou à défaut de s'assurer de la non-existence de celle-ci. Ceci lui permettra de prendre des décisions avec la certitude des résultats obtenues par l'algorithme d'optimisation. Ceci représente des gains substantiels en termes de temps et de coûts de développement.

L'espace d'optimisation en pré-dimensionnement étant continu, la méthode de Monte-Carlo qui consiste à faire des tirages aléatoires dans l'espace des solutions permet d'avoir cette garantie recherchée mais en temps infini. En effet, il faut que la méthode ait un temps d'exécution convenable dans le contexte industriel. C'est une contrainte très importante qui fera l'objet de la contribution de ce chapitre.

L'arithmétique d'intervalles apparue en 1966 [Moore R.E, 1966] permet de voir le problème autrement. Cette arithmétique traite des intervalles au lieu de traiter des nombres. Ainsi, il devient possible de discrétiser un espace continu en un ensemble d'intervalles. Dans le cas multidimensionnel, on parle de boites.

Dans cette partie, nous considérons les équations de la fonction objectif et des contraintes analytiquement disponibles et continument différentiables. Les modèles avec des systèmes implicites et des fonctionnelles ne sont pas traités dans cette partie.

Les problèmes résolus dans cette partie sont les problèmes d'optimisation algébrique sous contraintes. Ils sont formulés de façon générale comme suit :

$$\begin{cases} min_{x \in X \subseteq \mathbb{R}^n} f(x) \\ g_k(x) \leq 0, k=1, \dots, p \\ h_k(x) = 0, k=p+1, \dots, p+q \end{cases} \quad (2.1)$$

où f est la fonction objectif, $\{g_k, k=1, \dots, p\}$ l'ensemble des contraintes d'inégalité, et $\{h_k, k = p+1, \dots, p+q\}$ l'ensemble des contraintes d'égalité. L'optimum est recherché dans le domaine $X \subseteq \mathbb{R}^n$ et doit respecter toutes les contraintes d'égalité et d'inégalité.

La méthode Interval Branch&Bound Algorithm (appelée IBBA par [Messine F. et al, 1998]) qui sera développée dans cette partie a deux objectifs :

- Trouver et encadrer l'optimum global et tous les optimiseurs avec une précision déterminée par l'utilisateur.
- Afficher des temps d'exécution raisonnables (nombre d'évaluations du modèle) dans la résolution de problèmes de pré-dimensionnement en ingénierie.

La méthode IBBA est basée sur l'arithmétique d'intervalles et la propagation de contraintes :

- Arithmétique d'intervalles : permet d'évaluer des fonctions sur des intervalles. Ceci permet de rendre possible la discrétisation de problèmes continus et l'utilisation des techniques de séparation évaluation (Branch & Bound). Ces techniques consistent à séparer l'espace de recherche en différentes parties (des boites), et éliminer les parties qui ne contiennent pas des solutions.
- Propagation de contraintes : permet de réduire le domaine de recherche de l'optimum en éliminant les régions non-faisables. Les contracteurs sont les outils en œuvre de la propagation de contraintes. Ils sont issus du domaine de la programmation par contraintes et CSP (Constraint Satisfaction Problem). Les contracteurs, comme leur nom l'indique, permettent de contracter une partie de l'espace de recherche de solution en éliminant les régions qui ne respectent pas les contraintes (régions non consistantes).

La contribution de ce chapitre consiste en une reformulation qui permet de rendre IBBA plus efficace en réduisant le nombre de variables de décision (techniques d'accélération). Cette reformulation utilise les caractéristiques intrinsèques de beaucoup de problèmes d'ingénierie, à savoir la distinction entre les paramètres d'entrées et de sorties. L'idée principale est de simplifier au maximum le problème afin de réduire le nombre d'évaluations du modèle. Ceci devient possible en traitant uniquement les paramètres indépendants du modèle.

Afin d'introduire l'algorithme IBBA, les notions de bases de l'arithmétique d'intervalles seront introduites dans le chapitre 2.1. Dans le paragraphe 2.2.1, un algorithme IBBA pour résoudre des problèmes non contraints sera présenté. Ensuite, les contracteurs issus de la propagation de contraintes seront introduits dans le paragraphe 2.2.2. Le paragraphe 2.3 sera dédié à la présentation de la reformulation ainsi que les tests numériques. Enfin, le paragraphe 2.4 fera l'objet de la conclusion du chapitre.

2.1. L'arithmétique d'intervalles : notions de base

L'algorithme d'optimisation globale IBBA est basé sur le principe de séparation évaluation. Les problèmes étant continus, il est nécessaire de discrétiser le domaine de recherche continu afin de diviser le domaine de recherche. L'arithmétique d'intervalles a été introduite par R.E Moore en 1966 comme moyen pour maîtriser les problèmes d'arrondi dans les calculs à haute précision [Moore R.E, 1966]. Depuis 1966, plusieurs développements dans l'optimisation globale déterministe ont vu le jour prenant avantage de la puissance croissante des ordinateurs [Hansen E., Walster G. W, 2004] [Voller R. L. et al, 1991] [Kearfott R. B. et al, 1991] [Messine F., 2004] [Faltings B. et al, 2001].

Définition (Intervalle) : Un intervalle est un sous-ensemble de $\mathbb{R}$ *connecté ([1,2]U[3,5] n'est pas un intervalle), fermé ([1,2[n'est pas un intervalle). L'ensemble de tous les intervalles de* $\mathbb{R}$ *sera noté* $\mathbb{IR}$.

Définition (Boite) : Une boite est un produit cartésien de n intervalles c'est à dire un vecteur d'intervalles. ([1,2],[4,5],[4,10]) *est une boite tridimensionnelle. Une boîte à n dimensions est un élément de* $\mathbb{IR}^n$. *Une boite est appelée aussi « hypercube ».*

2.1.1. Opérations élémentaires

Considérons deux intervalles I_1 et I_2. On définit :

$I_1 \cap I_2 = \{x/\ x \in I_1 \wedge x \in I_2\}$

$I_1 \cup I_2 = \{x/\ x \in I_1 \vee x \in I_2\}$

$I_1 \backslash I_2 = \{x/\ x \in I_1 \wedge x \notin I_2\}$

$I_1 \times I_2 = \{(x,y)/\ x \in I_1 \wedge y \in I_2\}$

Définition (Hull): L'opérateur hull appliqué à un ensemble de $\mathbb{R}^n$ *correspond à la plus petite boite contenant cet ensemble.*

$$\forall D \in \mathcal{P}(\mathbb{R}^n),\quad (D) = inf_{\subseteq}\big(X, (X \in \mathbb{IR}^n) \wedge (D \subseteq X)\big) \tag{2.2}$$

avec $\mathcal{P}(\mathbb{R}^n)$ *l'ensemble des parties de* $\mathbb{R}^n$.

Définition (Image): Soit f *une fonction de* $D \subseteq \mathbb{R}^n$ *dans* $\mathbb{R}^m$. *Soit* $X \in \mathbb{I}D$. *On appelle image de X par la fonction* f *le sous-ensemble suivant :*

$$Image(f, X) = \{f(x)/x \in X\} \tag{2.3}$$

Au lieu de définir les opérateurs sur les réels (arithmétique classique), l'arithmétique d'intervalles étend l'utilisation de ces opérateurs aux intervalles. Pour cela, tous les opérateurs élémentaires ont été redéfinis pour réaliser les opérations sur des intervalles :

Définition (opérateurs élémentaires) : Soient X et Y deux intervalles et $\circ \in \{+, -, \times, \div\}$*:*

$$X \circ Y = \{x \circ y \,/\, x \in X, y \in Y\} \tag{2.4}$$

Par conséquent :

$[a, b] + [c, d] = [a + c, b + d]$

$[a, b] - [c, d] = [a - d, b - c]$

$[a, b] \times [c, d] = [min(a.c, a.d, b.c, b.d), max(a.c, a.d, b.c, b.d)]$

$[a, b] \div [c, d] = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)]$ et $0 \notin [c, d]$

La division est initialement prévu uniquement pour le cas $0 \notin [c, d]$. D'autres extensions ont été proposées pour prendre en compte le cas $0 \in [c, d]$ [Hansen E., Walster G. W., 2004].

Les fonctions élémentaires (*cosinus*, *sinus*, *logarithme* …) ont également été étendues aux intervalles. Voici l'exemple de la fonction cosinus :

Soit I l'intervalle $[a, b]$. On définit $e_1 = E(a/\pi)$ et $e_2 = E(b/\pi)$ avec E la fonction qui renvoi l'arrondi inférieur d'un réel :

$$cos(I) = \begin{cases} [-1,1] & si\ e1 < e2 - 1 \\ [min(cos(a), cos(b)), 1] & si\ e1 = e2 - 1\ et\ e1\ impaire \\ [-1, max(cos(a), cos(b))] & si\ e1 = e2 - 1\ et\ e1\ paire \\ [cos(a), cos(b)] & si\ e1 = e2\ et\ e1\ impaire \\ [cos(b), cos(a)] & si\ e1 = e2\ et\ e1\ paire \end{cases} \tag{2.5}$$

On définit les notions suivantes pour les intervalles et les boites :

Soit *I* l'intervalle $[a, b]$:

- $Mid(I)$: le milieu de l'intervalle définit par $(a + b)/2$. Pour une boite, c'est le vecteur des milieux des intervalles.
- $W(I)$: la largeur de l'intervalle définit par $b - a$. Pour une boite, c'est la largeur de l'intervalle le plus large. Par exemple, pour la boite $X = ([1,2], [4,5], [4,10])$, $W(X) = 6$.

Une opération particulièrement importante pour l'algorithme IBBA est la bissection puisque c'est celle-ci qui permet de discrétiser l'espace de recherche de l'optimum. Cet opérateur vise

à scinder un intervalle I (respectivement une boite B) en deux intervalles I_1 et I_2 tels que $I = I_1 \cup I_2$ (respectivement en deux boites B_1 et B_2 telles que $B = B_1 \cup B_2$).

La bissection d'un intervalle $[a, b]$ donne deux intervalles $[a, (a+b)/2]$ et $[(a+b)/2, b]$. La bissection d'une boite n'est pas aussi simple. En effet, elle nécessite le choix d'une direction de coupe. La règle la plus utilisée pour choisir cette direction est basée sur la largeur. En effet, elle préconise de couper la boite selon la direction la plus large. Par exemple, la bissection de la boite $X = ([1,2], [4,5], [4,10])$ selon sa plus large direction donne $X_1 = ([1,2], [4,5], [4,7])$ et $X_2 = ([1,2], [4,5], [7,10])$.

Il existe d'autres règles pour la bissection des boites mais dans la suite de ce manuscrit nous utiliserons la bissection selon la direction la plus large.

2.1.2. Fonctions d'inclusion

Les contraintes qui sont traitées dans les modèles analytiques d'ingénierie sont des combinaisons des opérations de base et des fonctions élémentaires. Afin de pouvoir les manipuler et les évaluer sur des intervalles, les extensions des fonctions plus complexes[1] sont introduites dans cette section.

Les fonctions d'inclusion permettent d'étendre n'importe quelle fonction aux intervalles. Cette propriété est particulièrement intéressante dans la propagation des contraintes.

Définition (Image) : Soit f une fonction de $D \subseteq \mathbb{R}^n$ dans $\mathbb{R}^m$, soit $X \in \mathbb{I}D$. On appelle Image de X par f le sous-ensemble suivant :

$$Image(f, X) = \{f(x)/x \in X\} \tag{2.6}$$

Définition (fonction d'inclusion) : Soit f une fonction de $D \subseteq \mathbb{R}^n$ dans $\mathbb{R}^m$. Soit F une autre fonction de $\mathbb{I}D$ in $\mathbb{I}\mathbb{R}^m$. F est appelée fonction d'inclusion de f dans D si et seulement si:

$$Image(f, X) \subseteq F(X) \tag{2.7}$$

Plus une fonction d'inclusion est proche de $Image(f, X)$ mieux c'est car elle donnera une évaluation plus précise.

[1] Complexes par rapports aux fonctions élémentaires. Ici on parle des fonctions usuelles qui sont des combinaisons des opérateurs élémentaires (+,-,*…) et des fonctions élémentaires (log, exp, sin …).

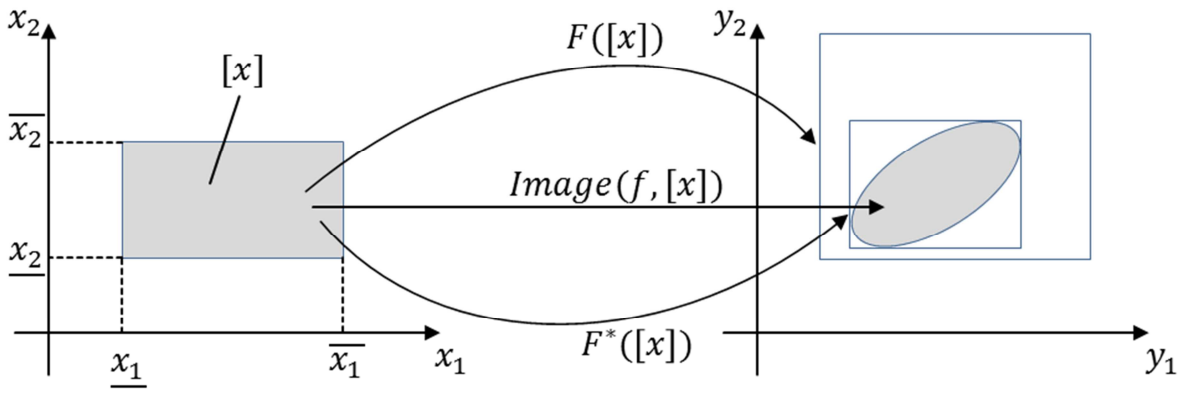

Figure 2-1. Fonctions d'inclusion F et F^* : F^* est minimale [Chabert G. et al, 2009]

Dans la Figure 2-1, F est une fonction d'inclusion et F^* la fonction d'inclusion minimale car $F^*([x]) = (Image(f,[x]))$.

Sachant que les extensions des opérateurs et fonctions élémentaires sont par définition des fonctions d'inclusion de celles-ci, une fonction d'inclusion triviale peut être construite. Celle-ci est appelée fonction d'inclusion naturelle. Elle est obtenue en inter-changeant les opérateurs classiques dans la fonction usuelle par leurs extensions aux intervalles.

Définition (Extension naturelle): Soit $f: \mathbb{R}^n \to \mathbb{R}$ *une fonction continue. La fonction* $F: \mathbb{IR}^n \to \mathbb{IR}$ *ayant la même expression que* f*, dans laquelle les opérateurs classiques sont remplacés par leurs extensions aux intervalles est appelée fonction d'inclusion naturelle de* f.

Prenons le cas d'une fonction simple :

$$f: \mathbb{R} \to \mathbb{R}$$

$$x \mapsto x^2 - 2x + 1$$

La fonction d'inclusion naturelle de f a la même expression mais utilisant les extensions aux intervalles des opérateurs :

$$F_1: \mathbb{IR} \to \mathbb{IR}$$

$$X \mapsto X^2 - 2X + 1$$

Pour $X = [1,2]$:

$$F_1(X) = [1,2]^2 - 2 * [1,2] + 1 = [-2,3]$$

Maintenant, si on utilise la même fonction mais sous une autre forme :

$$F_2: \mathbb{IR} \to \mathbb{IR}$$

$$X \mapsto (X-1)^2$$

Cette fonction d'inclusion est une autre extension de f. Pourtant :

$$F_2(X) = ([1,2] - 1)^2 = [0,1] = Image(f, X)$$ (Figure 2-2)

Par conséquent, on peut dire que la qualité de l'évaluation par intervalle d'une fonction dépend fortement de la fonction d'inclusion utilisée.

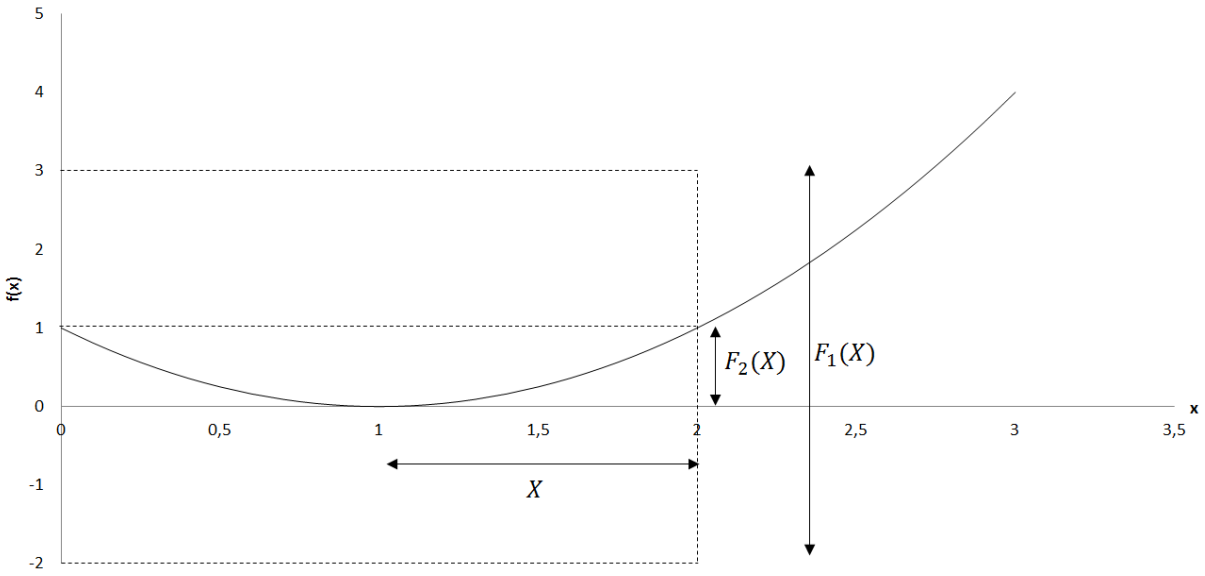

Figure 2-2. L'évaluation d'une fonction par deux fonctions d'inclusion différentes

Plus la fonction d'inclusion est précise, plus l'algorithme d'optimisation est efficace étant donné qu'il traite des domaines de recherche plus petits. C'est pour cette raison que plusieurs fonctions d'inclusion ont été introduites comme les formes centrées : Taylor au premier ordre, formes centrées optimales, Kite … etc [Baumann E., 1988] [Tamas T. C., 1991].

Définition (Extension de Taylor au premier ordre): Soit $f: D \subseteq \mathbb{R}^n \to \mathbb{R}^m$ *une fonction continument différentiable. Soit* $X \in \mathbb{I}D$ *et* J *l'extension aux intervalles du Jacobien de* f*. Soit* $x \in X$ *(par exemple le milieu). On peut démontrer par le théorème de Taylor et la formule du reste de Taylor-Cauchy que :*

$$Image(f, X) \subseteq f(x) + J(X)(X - x) \qquad (2.8)$$

Par conséquent, la fonction $f_t: \begin{matrix} \mathbb{I}D \to \mathbb{R}^m \\ X \mapsto f(x) + J(X)(X - x) \end{matrix}$ *est une extension aux intervalles de* f *et est appelée l'extension de Taylor du premier ordre.*

L'extension de Taylor est un cas particulier des formes centrées. Ces extensions s'appelles formes centrées car elles utilisent tous un point de X qui est souvent $\mathrm{Mid}(X)$.

L’extension de Taylor du premier ordre, tout comme les formes centrées, permet de linéariser la fonction f quel que soit son niveau de non-linéarité. Cette propriété peut s’avérer très utile dans la propagation de contraintes qui sera développée plus loin dans ce manuscrit.

Pour plus de détails sur l’arithmétique d’intervalles voir [Moore R.E, 1966] [Kearfott R.B, 1996].

2.2. Optimisation globale déterministe

Les algorithmes déterministes d'optimisation globale par intervalles utilisent le principe de la séparation-évaluation : la séparation est effectuée en divisant le domaine de recherche (hypercube) tandis que l'évaluation correspond à l'évaluation par les fonctions d'inclusion de la fonction objectif et des contraintes. Ensuite, les boites qui ne contiennent pas l'optimum sont écartées. Le premier algorithme introduit traite des problèmes non-contraints :

$$min_{x \in X \subseteq \mathbb{R}^n} f(x) \tag{2.9}$$

Ensuite, les techniques de propagation de contraintes issues de la programmation par contraintes ont été adaptées aux intervalles afin de traiter les contraintes.

L'algorithme d'optimisation général maintient une borne supérieur $\tilde{f}$ du minimum global x^* et la met à jour au fur et à mesure des itérations. Cette borne vérifie la propriété :

$$\tilde{f} > x^* \tag{2.10}$$

La borne supérieure du minimum global du problème d'optimisation traité est mise à jour via une solution réalisable. En effet, un test qui sera introduit permet de prendre une solution dans une boite du domaine de recherche. La borne supérieure n'est mise à jour que si ce point est faisable ET est meilleur que l'ancienne borne supérieure. La borne supérieure correspond donc toujours à une solution réalisable du point de vue des contraintes et c'est cette borne qui sera affinée au cours des itérations et qui sera considérée comme solution optimale avec une précision ϵ.

Si l'on considère un hypercube X tel que $F(X) > \tilde{f}$, c'est-à-dire que la borne inférieure de $F(X)$ est supérieure à $\tilde{f}$. Dans ce cas, la boite X ne peut pas contenir le minimum global et par conséquent est éliminée. A chaque itération, le modèle est évalué afin de calculer la borne inférieure de la fonction objectif et d'évaluer les contraintes sur la boite courante.

L'Algorithme 2-1 est l'algorithme d'optimisation par intervalles sans contraintes. Il prend en entrée une boite qui représente le domaine où l'optimum est recherché, et une précision ϵ. En sortie, il génère une liste de boites $resultList$ qui contient nécessairement l'optimum global, si celui-ci existe, et une borne supérieure $\tilde{f}$. L'Algorithme 2-1 maintient deux listes de boites au cours de son exécution. La première liste $workList$ contient les boites qui doivent être évaluées et vérifiées, ce sont les boites où l'inexistence de l'optimum est incertaine. La deuxième liste $resultList$ contient les boites qui n'ont pas été éliminées et dont la largeur ne dépasse pas une précision ϵ fixée par l'utilisateur.

Pour chaque boite dans la liste $workList$, on associe une borne inférieure de la fonction objectif. Cette liste peut être triée par ordre croissant des bornes inférieures. Ceci permet d'avoir de meilleures performances.

D'autres façons d'éliminer des boites existent et seront développées plus loin dans ce manuscrit.

Algorithme 2-1. Algorithme d'optimisation globale basique

1: Initialize X← The initial box in which the global minimum is sought.

2: Initialize $\tilde{f} \leftarrow +\infty$

3: Initialize the list $workList \leftarrow (+\infty, X)$: The structure that will contain the boxes that remain to be explored. The elements in the $workList$ have the form: (lower bound of f, Box).

4: Initialize the list $resultList$: $empty$

While ($workList$ not empty)

5: Take an element (v_j, X_j) from workList

6: Bisect X_j into two boxes: $X_{j1}\ and\ X_{j2}$

For i from 1 to 2

7: Compute the lower bound of f in X_{ji} : b_{inf}

If $(b_{inf} > \tilde{f})$

8: Discard X_{ji}

Else

9: $c \leftarrow Center(X_{ji})$

If $(f(c) < \tilde{f}\ and\ c\ feasible)$

10: $\tilde{f} \leftarrow f(c)$

End If

If $(w(X_{ji}) < \epsilon)$

11: $resultList \leftarrow resultList + (b_{inf}, X_{ji})$

Else

12: $workList \leftarrow workList + (b_{inf}, X_{ji})$

End If

End If

End For

End While

2.2.1. Optimisation globale : problèmes non-contraints

Les principaux algorithmes pour l'optimisation globale sans contraintes ont été développés par Eldon Hansen. Il s'est focalisé sur le cas unidimensionnel [Hansen E., 1979]. Ensuite, il a proposé une extension pour le cas multidimensionnel [Hansen E., 1980].

L'Algorithme 2-2 reprend les grandes lignes de l'Algorithme 2-1 en lui ajoutant deux tests. Ces améliorations, appelées aussi techniques d'accélération, ont grandement amélioré l'efficacité de cet algorithme le rendant ainsi plus pratique et plus rapide à converger. Une technique d'accélération est un test qui permet d'écarter des boites qui ne contiennent pas l'optimum global. Les deux techniques d'accélération ajoutées dans l'Algorithme 2-2 sont les tests de monotonie et de convexité [Voller R. L., 1991] [Michèle A. H., 1983].

L'idée principale du test de monotonie et d'écarter les boites dans lesquelles la fonction objectif est strictement monotone. Ce test est issu de la condition d'optimalité du premier ordre.

Condition d'optimalité du premier ordre : Soit f une fonction $f: A \to \mathbb{R}$ avec $A \subseteq \mathbb{R}^n$ ayant un optimum global $x^* \in A$. Si les dérivées partielles du premier ordre de f sont définies en x^* alors nécessairement :

$$G(f, x^*) = 0 \tag{2.11}$$

Avec G l'opérateur gradient.

Le test de convexité utilise le fait qu'au voisinage de l'optimum global, la fonction objectif est nécessairement convexe. Par conséquent, si nous avons la garantie que pour une boite donnée, la fonction objectif n'est pas convexe alors on peut affirmer que le minimum global ne s'y trouve pas. Ce test est issu de la condition d'optimalité du second ordre :

Condition d'optimalité du second ordre : Soit f une fonction $f: A \to \mathbb{R}$ avec $A \subseteq \mathbb{R}^n$ ayant un optimum global $x^* \in A$. Si f est deux fois dérivable en x^* alors nécessairement :

$$G(f, x^*) = 0 \text{ et } H(f, x^*) \geq 0 \tag{2.12}$$

Avec G et H respectivement les opérateurs gradient et hessian.

Algorithme 2-2. Unconstrained Global Optimization Algorithm

1: Initialize X← The initial box in which the global minimum is sought.

2: Initialize $\tilde{f} \leftarrow +\infty$

3: Initialize the list $workList \leftarrow (+\infty, X)$: The structure that will contain the boxes that remain to be explored. The elements in the $workList$ have the form: (lower bound of f, Box).

4: Initialize the list $resultList$: $empty$

While ($workList$ not empty)

5: Take an element (v_j, X_j) from workList

6: Bisect X_j into two boxes: $X_{j1}\ and\ X_{j2}$

For i from 1 to 2

7: Compute the lower bound of f in X_{ji} : b_{inf}

If ($b_{inf} > \tilde{f}$)

8: Discard X_{ji}

Else

9: $C \leftarrow Center(X_{ji})$

If ($f(C) < \tilde{f}$)

10: $\tilde{f} \leftarrow f(C)$

End If

If (f is strictly monotonous)

11: Discard X_{ji}

Else

If(f is not convex)

12: Discard X_{ji}

Else

If ($w(X_{ji}) < \epsilon$)

13: $resultList \leftarrow resultList + (b_{inf}, X_{ji})$

Else

14: $workList \leftarrow workList + (b_{inf}, X_{ji})$

End If

End If

End If

End For

End While

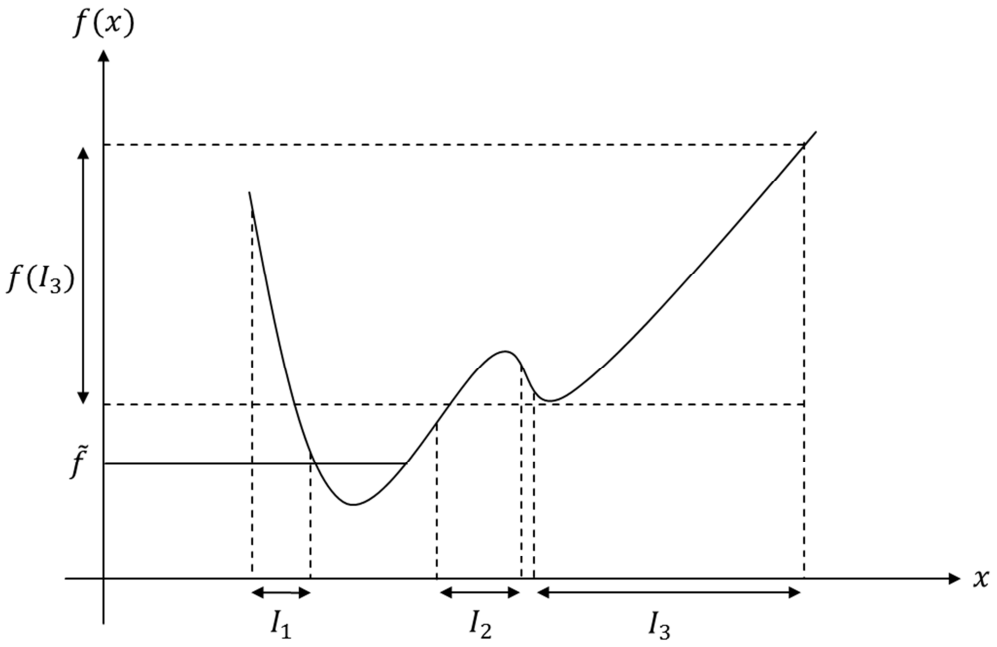

Figure 2-3. Les techniques d'accélération pour le cas non contraint

Prenons une fonction à une variable f dont nous cherchons le minimum global. Soient I_1, I_2 et I_3 trois intervalles de la liste $workList$. Ces intervalles ainsi que la fonction sont représentés dans la Figure 2-3. $\tilde{f}$ étant la dernière borne supérieure mise à jour du minimum global. Les trois intervalles I_1, I_2 et I_3 peuvent être éliminés par les techniques d'accélération de l'algorithme IBBA. En effet, la fonction f est trictement monotone dans I_1 qui peut être éliminé grâce à la condition d'optimalité du premier ordre. De même, I_2 peut être éliminé grâce à la condition d'optimalité du second ordre. Enfin, I_3 peut être éliminé car la fonction y est supérieure à $\tilde{f}$.

L'Algorithme 2-2 a été testé avec et sans les techniques d'accélération sur la fonction suivante (Six-hamp camel back) :

$$f_8: \begin{matrix} \mathbb{R}^2 \to \mathbb{R} \\ (x_1, x_2) \mapsto 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \end{matrix}$$

Le domaine de recherche initial est [-1000, 1000]² :

$$\min_{x \in [-1000,1000]^2 \subseteq \mathbb{R}^2} f_8(x)$$

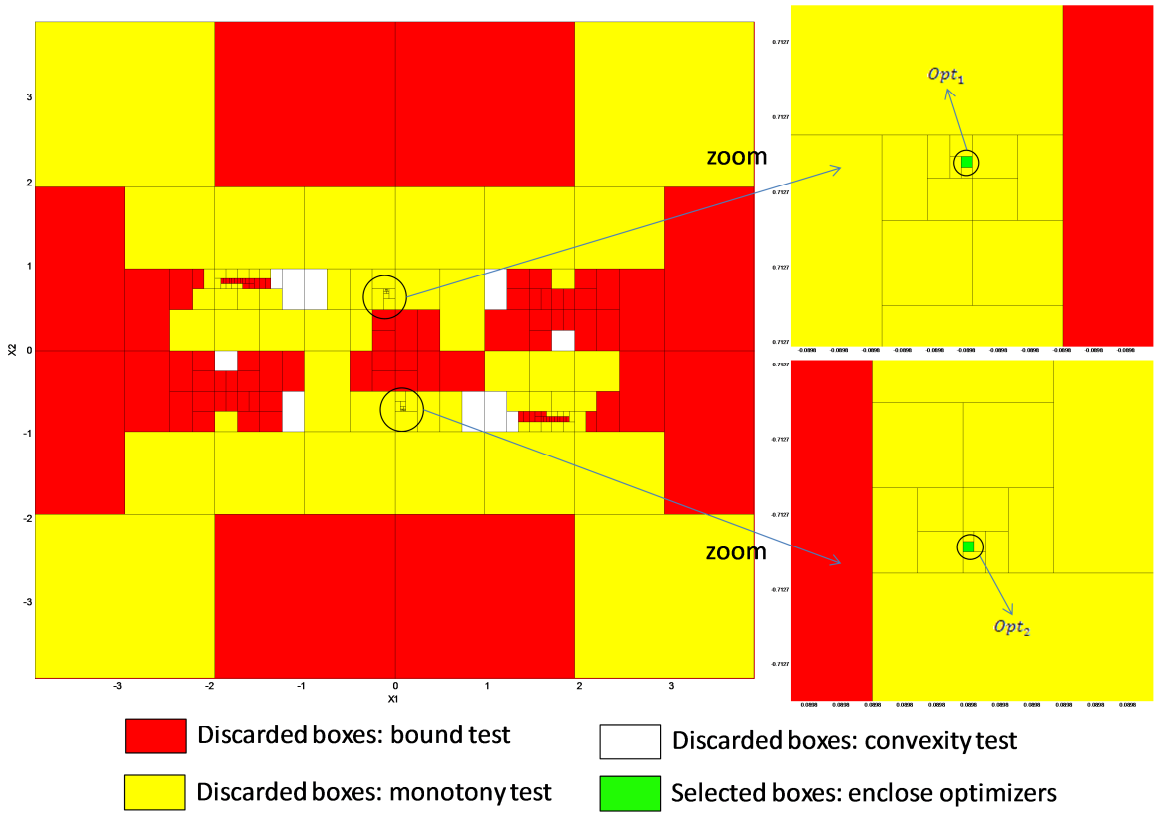

Figure 2-4. Zoom sur le pavage du domaine de recherche au voisinage de l'optimum (f_8)

Le Tableau 2-1 résume les principaux résultats obtenue avec l'Algorithme 2-2 sur la fonction f_8 avec et sans techniques d'accélération. En utilisant les techniques d'accélération, la borne supérieure de l'optimum global obtenue est -1.0316 et les optimiseurs ont été encadrés par deux boites avec une précision ϵ=10^{-8} :

$$Opt_1 = \begin{pmatrix} [-8.9842018e-2; -8.9842011e-2] \\ [7.12656401e-1; 7.12656408e-1] \end{pmatrix}$$

$$Opt_2 = \begin{pmatrix} [8.9842011e-2; 8.9842018e-2] \\ [-7.12656408e-1; -7.12656401e-1] \end{pmatrix}$$

Tableau 2-1. Impact des techniques d'accélération sur IBBA

f_8	Borne supérieure	Itérations (#)	Durée (s)
Sans techniques d'accélération	-1.0312	1 464 111	17.4
Avec techniques d'accélération	-1.0316	667	0.15

Durant l'exécution de l'Algorithme 2-2 avec les techniques d'accélération, 150 boites ont été écartées avec le test de la borne supérieure (en rouge), 182 avec le test de monotonie (en jaune) et 2 boites contenant les optimiseurs ont été retenues (en vert) (voir Figure 2-4).

Les résultats du Tableau 2-1 montrent l'intérêt de l'utilisation des techniques d'accélération dans le cas non contraint. La borne supérieure du minimum est obtenue avec une précision de 10^{-8}.

Ces techniques sont inadaptées dans le cas de l'optimisation avec contraintes. En effet, prenons un exemple simple de problème d'optimisation contraint :

$$\begin{aligned} &min\, f(x) = x^2 + 5x \\ &st: g(x) = 5x + 25 \leq 0 \\ &x \in \mathbb{R} \end{aligned} \qquad (2.13)$$

La contrainte restreint le domaine de recherche. En effet $5x + 25 \leq 0$ implique que x doit être inférieur à 5. Le domaine de recherche est par conséquent réduit à $[-\infty, -5]$. On peut facilement vérifier que la fonction objectif est monotone sur cet intervalle, ce qui signifie qu'à l'optimum global le gradient est nécessairement non nul. Cet exemple illustre le fait que le test de monotonie est inapproprié en présence de contraintes. De façon similaire, le test de convexité est inadapté en optimisation sous contraintes.

D'autres techniques ont été intégrées pour accélérer la résolution de problèmes contraints. Ces techniques feront l'objet du prochain paragraphe.

2.2.2. Optimisation globale : problèmes contraints

Les techniques d'accélération en présence de contraintes sont issues de la programmation par contraintes, plus précisément des techniques de propagation de contraintes.

Définition : un Problème de Satisfaction de Contraintes, en anglais CSP (Constraint Satisfaction Problem) est un triplet $< X, D, C >$ *défini par :*

- $X = \{x_1, x_2 \dots x_n\}$ *: un ensemble de* n *variables.*
- $D = \{D_1, D_2 \dots D_n\}$ *: l'ensemble des domaines des variables tels que* $x_1 \in D_1, x_2 \in D_2 \dots, x_n \in D_n$. D_i *peut être continu ou discret.*
- $C = \{C_1, C_2 \dots C_m\}$ *: un ensemble de* m *contraintes sur les variables.*

Pour chaque contrainte C_i, on note par le symbole ρ_{C_i}l'ensemble de tous les points de $\mathbb{R}^n$ qui respectent la contrainte.

Des techniques d'accélération basées sur la propagation de contraintes ont été développées afin d'écarter les régions non faisables du domaine de recherche. Ceci nous conduit à la notion de contracteur : une fonction qui permet d'éliminer les régions non consistantes (non faisables) du domaine de recherche.

Définition (Contracteur) : un contracteur, appelé aussi opérateur de rétrécissement (en anglais narrowing operator), pour une contrainte C *est une fonction* $C_{\mathbb{x}}: \mathbb{IR}^n \longrightarrow \mathbb{IR}^n$ *qui satisfait les deux propriétés suivantes :*

$$\forall x \in \mathbb{IR}^n, \begin{cases} C_{\mathbb{x}}(x) \subseteq x \ \ (contractance) \\ C_{\mathbb{x}}(x) \cap \rho_c = x \cap \rho_c (complétude) \end{cases} \tag{2.14}$$

Prenons l'exemple de la contrainte $C: x_1 = x_2 + x_3$ et les domaines D_1, D_2 et D_3 respectifs de x_1, x_2 et x_3. Un opérateur de rétrécissement possible peut être formulé comme suit :

- $D_1 = D_1 \cap (D_2 + D_3)$
- $D_2 = D_2 \cap (D_1 - D_3)$
- $D_3 = D_3 \cap (D_1 - D_2)$

Une bibliothèque a été implémentée pour mettre en place des opérateurs de rétrécissement pour les contraintes simples. Voici l'exemple de la contrainte moins ($x = y - z$) :

MinusContractor(x, y, z)

1. $x = x \cap (y - z)$
2. $y = y \cap (x + z)$
3. $z = z \cap (y - x)$

Et ici l'exemple de la contraintes carré ($y = x^2$)

SqrContractor(x, y)

1. $x_l = x \cap]-\infty, 0]$
2. $x_r = x \cap [0, +\infty[$
3. $y_l = x \cap Sqr(x_l)$
4. $y_r = x \cap Sqr(x_r)$
5. $y = y_l \cup y_r$
6. $if\ y = \emptyset\ then\ x := \emptyset$
7. $else$
8. $x_l = x_l \cap [-sqrt(right(y)), -sqrt(left(y))]$
9. $x_r = x_r \cap [+sqrt(left(y)), +sqrt(right(y))]$
10. $x = x_l \cup x_r$

Les contraintes utilisées dans les opérateurs de contractions MinusContractor et SqrContractor sont simples. Pour les contraintes linéaires, il est possible de formuler facilement une fonction de contraction car il est possible d'exprimer chaque variable en fonction des autres. Ceci devient beaucoup plus difficile voire impossible dans le cas général des contraintes analytiques non linéaires. Pour les contraintes plus complexes, deux solutions ont été proposées :

- Une méthode basée sur la linéarisation et qui utilise les formes centrées [Hansen E., Walster G. W., 2004]. Pour une contrainte $C(x) = 0$ on peut calculer l'extension de Taylor au premier ordre :

$$\forall y \in X, C(X) \subseteq C(y) + C'(X)(X - y)$$

  Ensuite, on peut facilement exprimer chaque variable en fonction des autres :

$$X_i = X_i \cap \left(y_i + \frac{-C(y) - \sum_{j=1, j\neq i}^{n} C'_j(X)(X_j - Y_j)}{C'_i(X)}\right) \quad (2.15)$$

  avec X_i la i^{ème} composante du vecteur X. Ainsi, pour utiliser cette technique de contraction, il est nécessaire de calculer toutes les dérivées de la contrainte. Calculer à la main les dérivées peut être difficile pour des problèmes d'ingénierie avec plusieurs variables et plusieurs équations. Il est possible d'utiliser des techniques de différentiation automatique disponibles.

- Une méthode basée sur l'arbre de calcul [Benhamou F. et al, 1999] [Messine F., 2004]. Cette méthode consiste à développer chaque contrainte en créant des variables intermédiaires afin de transformer une contrainte complexe en une suite de contraintes élémentaires. Ensuite, chaque contrainte atomique est traitée via le contracteur correspondant [Messine F., 2004]. Voici l'exemple du traitement de la contrainte :

$$C(x) = 2x_2x_3 + x_1, (x_1, x_2, x_3) \in [1,3]^3$$

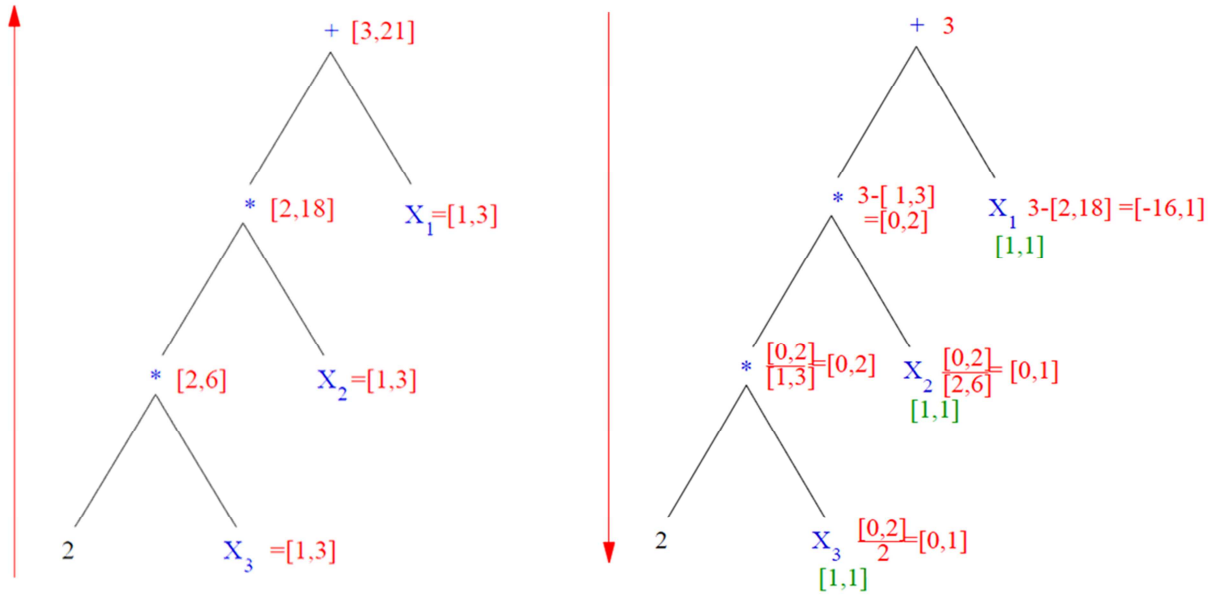

Figure 2-5. Utilisation de l'arbre de calcul pour la propagation de contraintes

En comparaison avec l'Algorithme 2-2 non contraint, deux principales modifications sont apportées dans l'Algorithme 2-3 :

- Chaque boîte prise dans la $workList$ est systématiquement contractée.
- Le critère d'arrêt contient une condition additionnelle : $\tilde{f} - b_{inf} < \epsilon_{bound}$. Ce test reflète une exigence sur la qualité de la solution car il nous garantit que le minimum global est supérieur à $\tilde{f} - \epsilon_{bound}$.

Des tests ont été réalisés pour évaluer la contribution des contracteurs :

Cas d'étude 1 :

$$\begin{aligned} &min\, f(x) = x_3 + (x_1 + x_2 + x_3)x_1x_4 \\ &st: \begin{cases} x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40 \\ x_1x_2x_3x_4 \geq 25 \end{cases} \\ &X \in [1,5]^4 \end{aligned} \tag{2.16}$$

Le minimum global : $f^* = 17.02, X^* = (1; 4.743; 3.821; 1.378)$

Cas d'étude 2 :

$$\begin{aligned} &min\, f(x) = -12x_1 - 7x_2 + x_2^2 \\ &st: 2x_1^4 + 2 - x_2 = 0 \\ &X \in [0,2] \times [0,3] \end{aligned} \tag{2.17}$$

Le minimum global : $f^* = -22.09075, X^* = (0.840; 2.999)$

Comme montré par le Tableau 2-2, sans l'utilisation de contracteurs, aucun des deux cas d'étude n'a été résolu. Ceci est causé par le nombre de boîtes de la $workList$ qui croit exponentiellement et qui dépasse la taille de la mémoire allouée par l'algorithme d'optimisation. Ceci est d'autant plus important que IBBA ne trouve pas de solution faisable donc n'arrive pas à mettre à jour la borne supérieure $\tilde{f}$. Par conséquent, aucune boite ne peut être éliminée par le test de la borne supérieure.

La bibliothèque de calcul par intervalle ainsi que les algorithmes d'optimisation ont été implémentés en Java.

Tableau 2-2. Tests numériques sur les cas d'étude 1 et 2

	Cas d'étude 1		Cas d'étude 2	
	Itérations (#)	Durée (ms)	Itérations (#)	Durée (ms)
Sans propagation	Non résolu		Non résolu	
Avec propagation	36 216	1500	49	9

Pour de plus amples détails sur les contracteurs voir [Benhamou F. et al, 1999] [Chabert G. et Jaulin L., 2009] [Araya I., Trombettoni G. et al, 2009].

Algorithme 2-3. Constrained Global Optimization Algorithm

1: Initialize X← The initial box in which the global minimum is sought.

2: Initialize $\tilde{f} \leftarrow +\infty$

3: Initialize the list $workList \leftarrow (+\infty, X)$: The structure that will contain the boxes that remain to be explored. The elements in the $workList$ have the form: (lower bound of f, Box).

4: Initialize the list $resultList$: $empty$

While (workList not empty)

5: Take an element (v_j, X_j) from workList

6: Bisect X_j into two boxes: $X_{j1}\ and\ X_{j2}$

For i from 1 to 2

7: Contract the box by constraint propagation techniques

8: Compute the lower bound of f in X_{ji} : b_{inf}

If $(b_{inf} > \tilde{f})$

9: Discard X_{ji}

Else

10: $C \leftarrow Center(X_{ji})$

If $(f(C) < \tilde{f}$ and C is feasible)

11: $\tilde{f} \leftarrow f(C)$

End If

If $(w(X_{ji}) < \epsilon_{width}\ and\ \tilde{f} - b_{inf} < \epsilon_{bound})$

12: $resultList \leftarrow resultList + (b_{inf}, X_{ji})$

Else

13: $workList \leftarrow workList + (b_{inf}, X_{ji})$

End If

End If

End For

End While

L’algorithme utilise le principe de séparation et évaluation. L’espace de recherche est divisé au fur et à mesure, l’arithmétique d’intervalles est utilisée pour évaluer les fonctions sur des boites et les contracteurs permettent de réduire ces boites.

Malgré l’efficacité des techniques d’accélération, spécialement la propagation de contraintes, la mise à jour de la borne supérieure du minimum global reste difficile. En effet, la borne supérieure est mise à jour en testant si le centre de la boite en cours de traitement est faisable. Ceci n’est pas facile si les contraintes sont difficiles. C’est le cas spécialement des contraintes égalité. En sachant que certaines variables sont reliées entre elles via les contraintes d’égalité, il est possible de réduire le nombre de variables de décision. Si on note par n_p le nombre de variables de décision et par n_e le nombre de contraintes d’égalité, supposées indépendantes, le problème peut être résumé en la détermination de $n_p - n_e$ paramètres. Les autres variables étant directement calculées via les contraintes d’égalité. Désormais, ces variables indépendantes sont appelées degrés de liberté. Cette simplification peut être atteinte au travers d’une reformulation qui sera expliquée dans le prochain paragraphe.

2.3. Contribution : la reformulation

2.3.1. La reformulation

Dans cette partie, une reformulation qui vise à améliorer les performances de l'algorithme IBBA sera présentée. L'idée de la reformulation vient d'un point faible de IBBA. Comme expliqué dans les paragraphes précédent, il y'a deux mécanismes qui impactent fortement l'efficacité de l'algorithme d'optimisation : l'étape de rétrécissement ou propagation de contraintes et l'étape de mise à jour de la borne. Les tests de l'algorithme IBBA sur plusieurs problèmes d'optimisation ont mis en évidence la difficulté de l'étape de mise à jour de la borne en présence de contraintes d'égalité. La reformulation a pour objectif de rendre cette étape plus facile et ainsi accélérer la convergence de l'algorithme. En effet, cela permet de se rapprocher plus rapidement du minimum global et d'écarter plus de régions de l'espace de recherche avec le test de la borne.

Afin de mieux comprendre la reformulation, regardons la structure d'un problème d'optimisation :

$$\begin{cases} min_{x \in X \subseteq \mathbb{R}^n} f(x) \\ g_k(x) \leq 0, k{=}1 \dots p \\ h_k(x) = 0, k{=}p+1 \dots p+q \end{cases}$$

Dans le domaine des mathématiques appliquées et de l'optimisation, on parle de paramètres du problème et de variables de décision. Dans la formulation présentée plus haut, x est le vecteur des variables de décision. C'est-à-dire l'ensemble de variables à instancier indépendamment afin d'optimiser la fonction objectif en respectant les contraintes.

Dans le monde de la simulation et de la conception de produits, on parle de paramètres du problème, de paramètres d'entrée et de paramètres de sortie. C'est-à-dire que les variables de décision du point de vue des mathématiciens sont divisées en deux catégories dans le monde des ingénieurs. En réalité, les ingénieurs exploitent le fait que les variables de décision ne sont pas réellement indépendantes mais sont liées entre elles par les équations de la physique. On parle ainsi de degrés de liberté qui représentent les paramètres indépendants du modèle.

Dans la pratique classique de l'algorithme IBBA, les équations physiques qui sont des égalités sont traitées comme des contraintes d'égalité. Prenons l'exemple de l'équation du couple électromagnétique du moteur présenté au début du manuscrit :

$$\Gamma_{em} = \frac{\pi}{2\lambda}(1 - K_f)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e$$

Dans ce cas, l'algorithme cherche à instancier les variables de cette équation physique afin de vérifier la contrainte :

$$\frac{\pi}{2\lambda}(1 - K_f)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e - \Gamma_{em} = 0$$

Cette formule est certes mathématiquement équivalente à la première mais elle n'exploite pas le lien entre les paramètres d'entrée ($\lambda, K_f, k_r, \beta, E_{ch}, E, D, B_e$) et les paramètres de sortie (Γ_{em}). Au lieu de l'exploiter, IBBA subit ce lien et essai de le vérifier en instanciant les paramètres indépendamment. Ceci complexifie grandement la convergence de l'algorithme IBBA. En pratique, il est toujours nécessaire de relâcher les contraintes d'égalité. La contrainte sur le couple du moteur est relâchée comme suit :

$$\frac{\pi}{2\lambda}\left(1 - K_f\right)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e - \Gamma_{em} \in [0 - \epsilon, 0 + \epsilon]$$

Sachant que l'algorithme IBBA est fait à la base pour donner un optimum garantie, cette relaxation lui fait perdre cette garantie.

L'idée de la reformulation arrive ainsi comme une évidence, au lieu de traiter toutes les variables du modèle indépendamment dans IBBA, pourquoi ne pas traiter uniquement les degrés de liberté qui représentent les variables physiquement indépendantes.

Un problème d'optimisation en conception est donc réécrit sous la forme suivante :

$$\begin{cases} min_{x \in X \subseteq \mathbb{R}^n} f(x) \\ \quad x_k = e_k(x), k = 1 \dots r \\ \quad y_k = f_k(x), k = 1 \dots p + q - r \end{cases} \tag{2.18}$$

où :

- $x_k, k = 1 \dots r$ sont les paramètres de sorties qui peuvent être calculés à partie des $(n - r)$ paramètres d'entrée via un sous-ensemble des contraintes initiales.
- $y_k = f_k(x)$: pour les contraintes d'égalité qu'il n'est pas possible de reformuler et les contraintes d'inégalité, de nouvelles variables y_k sont introduites de sorte à transformer les contraintes d'inégalité et le restant des contraintes égalité en contraintes d'égalité exclusivement. Donc, $y_k \in]-\infty, 0]$ si la contrainte est initialement une contrainte d'inégalité, et $y_k \in [0,0]$ si c'est une contrainte d'égalité.

Avec cette reformulation, l'algorithme IBBA ne traite que le vecteur de dimension $(n - r)$ des paramètres libres $x_{r+1} \dots x_n$. La détermination de ces variables est suffisante pour déterminer le restant des variables $x_1 \dots x_r$ et $y_1 \dots y_{p+q-r}$. Les contraintes reformulées doivent être adressées dans un ordre spécifique afin que le calcul puisse être effectué.

Cette reformulation est rigoureusement équivalente à la formulation initiale et aucune simplification n'est effectuée. Ainsi, on ne perd pas la garantie.

Considérons le modèle suivant :

$$\begin{cases} \min\limits_{P \in P_0} f(P) \\ sc: P_1 = h_1(P) \\ \quad P_2 = h_2(P) \\ \quad P_3 = h_3(P) \\ \quad P_4 = h_4(P) \end{cases}$$

avec $P = (P_1, P_2, P_3, P_4, P_5, P_6)$ le vecteur des paramètres de conception. Etant donné qu'il n'y a pas de systèmes implicites, les paramètres peuvent être classés en paramètres d'entrée et paramètres de sortie. Dans cet exemple, les paramètres de sortie sont P_1, P_2, P_3 et P_4. Les paramètres d'entrée sont P_5 et P_6. Concrètement cela signifie que pour une instanciation des paramètres d'entrée, les paramètres de sortie peuvent être calculés via les contraintes. Pour cela, les contraintes sont traitées dans un ordre spécifique afin de pouvoir faire ce calcul. Dans cet exemple, supposons que l'ordre est h_1, h_2, h_3 puis h_4. Par conséquent, h_1 est nécessairement une fonction des entrées uniquement. h_2 est une fonction des entrées et de la sortie de la fonction h_1 et ainsi de suite. Ceci permet de calculer en cascade tous les paramètres de sortie en fonction uniquement des paramètres d'entrée.

La première étape de la reformulation consiste donc à classer les contraintes d'égalité dans le bon ordre. La Figure 2-6 résume cette étape. On note par I l'ensemble des entrées et par O celui des sorties. $O(h_i)$ représente donc les sorties de la fonction h_i. Dans cet exemple, $O(h_1) = \{P_1\}$.

Soit n_p le nombre total de paramètres et n_e le nombre de contraintes d'égalité. Etant donné que les contraintes sont indépendantes, le problème a $n_p - n_e$ degrés de liberté, ce qui correspond au nombre des entrées.

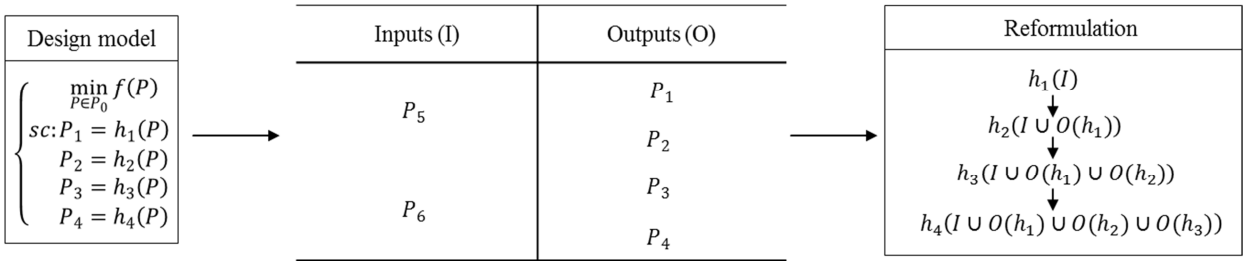

Figure 2-6. Illustration de la reformulation

La deuxième étape de la reformulation consiste à reformuler chaque contrainte de façon à mettre en entrée le maximum de paramètres fixes et en sortie le maximum de paramètres variables, ayant les domaines les plus larges en préférence. En effet, les contraintes d'intervalles sont beaucoup plus faciles à satisfaire que les contraintes d'égalité. Ceci n'est en théorie pas toujours possible mais dans des problèmes d'ingénierie c'est souvent le cas.

Etant donné que les sorties peuvent être calculées directement à partir de l'information sur les entrées, le problème d'optimisation est réduit à la détermination des entrées optimales tout

comme les problèmes de commande optimale. Par conséquent, IBBA peut ne traiter que la boite des entrées.

Cette formulation permet de classifier les variables du problème d'optimisation en paramètres d'entrée x et paramètres de sortie y. Pour la suite du manuscrit, on adopte cette formulation plus simple :

$$\begin{cases} min_{x \in X \subseteq \mathbb{R}^n} f(x) \\ y = g(x) \\ x \in [\underline{x}, \overline{x}], y \in [\underline{y}, \overline{y}] \end{cases} \quad (2.19)$$

Exemple. Considérons les contraintes suivantes d'un problème d'optimisation :

$$\begin{aligned} & g_1 : -x_4 + x_3 - 0.55 \leq 0 \\ & g_2 : -x_3 + x_4 - 0.55 \leq 0 \\ & h_1 : 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\ & h_2 : 1000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\ & h_3 : 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0 \\ & x_1 \geq 0, x_2 \leq 1200, x_3 \geq -0.55, x_4 \leq 0.55 \end{aligned} \quad (2.20)$$

Ce problème contient $n = 4$ variables, $p = 2$ contraintes d'inégalité et $q = 3$ contraintes d'égalité. Ces contraintes peuvent être reformulées et ordonnées comme suit :

$$\begin{aligned} & x_1 = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8, \\ & x_2 = 01000 \sin(x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8, \\ & y_1 = \ -x_4 + x_3 - 0.55 \leq 0, \\ & y_2 = -x_3 + x_4 - 0.55 \leq 0, \\ & y_3 = \ 1000 \sin(x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0, \\ & x_1 \geq 0, x_2 \leq 1200, x_3 \geq -0.55, x_4 \leq 0.55, \\ & y_1, y_2 \in]-\infty, 0], y_3 \in [0,0] \end{aligned} \quad (2.21)$$

Ainsi, l'algorithme d'optimisation traite le vecteur de dimension 2 des variables x_3 et x_4.

Deux étapes de l'algorithme IBBA doivent être revisitées afin de prendre en considération la reformulation : la bissection et la mise à jour de la borne. La bissection est réalisée uniquement sur les entrées. Par exemple si la boite en cours est :

([5,9];[1,5];[2,8];[3,4];[2,4];[1,2])

Sachant que les paramètres d'entrée sont x_5 et x_6, les boites résultantes sont :

([5,9];[1,5];[2,8];[3,4];[2,3];[1,2]) et ([5,9];[1,5];[2,8];[3,4];[3,4];[1,2])

Ici la bissection est effectuée suivant x_5 qui est la plus large parmi x_5 et x_6.

La mise à jour n'est plus faite par le centre de la boite en cours. Elle est réalisée plutôt par le centre de la boite des entrées et en calculant les sorties correspondantes ce qui permet de vérifier par construction les contraintes d'égalité. Les seules contraintes qui restent à vérifier sont donc les contraintes d'inégalité qui sont plus faciles à satisfaire.

Prenons l'exemple du cas d'étude 2 :

$$\begin{array}{l}\min f(x) = -12x_1 - 7x_2 + x_2^2 \\ st: x_2 = 2x_1^4 + 2 \\ X \in [0,2] \times [0,3]\end{array} \quad (2.22)$$

Ce problème est mathématiquement équivalent au problème initial. Ici, la sortie est x_2 et l'entrée x_1. Si la boite en cours de traitement est ([0,1],[0,2.5]), on prend le centre de la boite des entrées, ici 0.5, et on calcule les sorties qui correspondent, ici $x_2 = 2 \times 0.5^4 + 2 = 2.125$. Il est évident que le point calculé $(0.5\ ,\ 2.125)$ respecte par construction la contrainte d'égalité. Reste à vérifier si le point calculé appartient aux bornes. Ici il faut vérifier si 2.125 appartient à l'intervalle [0,3] ce qui est le cas dans cet exemple.

Une reformulation n'est souvent pas unique car elle dépend du choix des entrées et des sorties. Prenons l'exemple du moteur à aimants permanents.

$$min\, V_u = \pi \frac{D}{\lambda}(D + E - e - l_a)(2C + E + e + l_a)$$
$$s.c \begin{cases} h_1: \Gamma_{em} = \frac{\pi}{2\lambda}\left(1 - K_f\right)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e \\ h_2: E_{ch} = k_r E J_{cu}^2 \\ h_3: K_f = 1.5 p \beta \frac{e+E}{D} \\ h_4: B_e = \frac{2 l_a M}{D log\left[\frac{D+2E}{D-2(l_a+e)}\right]} \\ h_5: C = \frac{\pi \beta B_e}{4 p B_{fer}} D \\ h_6: p = \frac{\pi D}{\Delta_p} \end{cases} \quad (2.23)$$

Le Tableau 2-3 présente 4 formulations possibles : la formulation initiale et trois reformulation entrées/sorties. On introduit le coefficient de faisabilité statistique α. Ce coefficient est obtenu en calculant le pourcentage de solutions faisables parmi 106 solutions tirées aléatoirement dans l'espace de recherche initial. Ce coefficient représente la facilité de trouver une solution faisable et donc la difficulté des contraintes. On voit bien que selon la reformulation adoptée, le problème n'a pas la même « difficulté statistique ». Dans la reformulation initiale, aucune solution faisable n'a été obtenue parmi les 106 tirées aléatoirement. Avec les 3 autres reformulations, on trouve beaucoup plus de solutions faisables. Ces trois reformulations bien que n'ayant pas le même coefficient α, ont une faisabilité statistique très proche.

Actuellement, cette reformulation reste manuelle mais elle est tout à fait automatisable. En plus, elle demande que la formulation des équations le permette ce qui est souvent le cas.

Tableau 2-3. Les caractéristiques du modèle initial et des reformulations

Modèle	Entrées	Sorties	Ordre	α(%)
Initial	$D, \lambda, E \ldots$	Sans sorties	Pas d'ordre spécifique	0
Ref 1	l_a, E, β, e	$D, \lambda, C, B_e, J_{cu}, K_f$	$h_6, h_2, h_4, h_3, h_1, h_5$	5.11
Ref 2	l_a, C, J_{cu}, e	$D, \lambda, E, \beta, B_e, K_f$	$h_6, h_2, h_4, h_5, h_3, h_1$	4.82
Ref 3	l_a, B_e, K_f, e	$D, \lambda, E, C, \beta, J_{cu}$	$h_6, h_4, h_2, h_3, h_1, h_5$	4.34

2.3.2. Tests numériques sur les deux cas d'étude

Des tests numériques ont été réalisés afin de mettre en évidence l'intérêt de la reformulation et son impact sur les performances de l'algorithme IBBA. Dans un premier temps, des tests ont été effectués sur les deux cas d'étude présentés dans ce chapitre. Des améliorations notables ont été enregistrées en termes de nombre d'itérations et de durée d'exécution. Les résultats sont résumés dans le Tableau 2-4. En effet, on note une réduction du nombre d'itérations de 75% pour le cas d'étude 1 et de 40% pour le cas d'étude 2.

Cas d'étude 1

$$\min f(x) = x_3 + (x_1 + x_2 + x_3)x_1x_4$$
$$st: \begin{cases} x_1^2 + x_2^2 + x_3^2 + x_4^2 = 40 \\ x_1x_2x_3x_4 \geq 25 \end{cases} \quad X \in [1,5]^4$$

Cas d'étude 2

$$\min f(x) = -12x_1 - 7x_2 + x_2^2$$
$$st: 2x_1^4 + 2 - x_2 = 0 \;\; X \in [0,2] \times [0,3]$$

Reformulation

$$\min f(x) = x_3 + (x_1 + x_2 + x_3)x_1x_4$$
$$st: \begin{cases} x_1 = \sqrt{40 - x_2^2 - x_3^2 - x_4^2} \\ x_5 = x_1x_2x_3x_4 \\ X \in [1,5]^4 \times [25, +\infty[\end{cases}$$

Reformulation

$$\min f(x) = -12x_1 - 7x_2 + x_2^2$$
$$st: x_2 = 2x_1^4 + 2 \;\; X \in [0,2] \times [0,3]$$

Tableau 2-4. Tests numériques sur les cas d'étude 1 et 2 avec et sans reformulation

	Cas d'étude 1		Cas d'étude 2	
	Itérations (#)	Durée (ms)	Itérations (#)	Durée (ms)
Sans reformulation	36 216	1500	49	9
Avec reformulation	9396	300	29	7
Gain	74%	80%	41%	22%

Ceci s'explique par la réduction du nombre de variables de décision pour passer de 5 à 3 pour le cas d'étude 1 et de 2 à 1 pour le cas d'étude 2. Aussi, les contraintes d'égalité ont laissé la place à des contraintes d'inégalité beaucoup plus faciles à satisfaire. IBBA arrive donc à mettre à jour la borne supérieure du minimum global beaucoup plus facilement ce qui accélère sa convergence.

2.3.3. Conception optimale du moteur à aimants permanents

L'efficacité de la reformulation a été testée sur le modèle de conception du moteur à aimants permanents présenté dans le paragraphe 1.2. Le modèle initial du moteur est le suivant :

$$
\begin{aligned}
&min\, V_u = \pi \frac{D}{\lambda}(D + E - e - l_a)(2C + E + e + l_a) \\
&s.c \begin{cases}
\Gamma_{em} = \frac{\pi}{2\lambda}\left(1 - K_f\right)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e \\
E_{ch} = k_r E J_{cu}^2 \\
K_f = 1.5 p \beta \frac{e+E}{D} \\
B_e = \frac{2 l_a M}{D log\left[\frac{D+2E}{D-2(l_a+e)}\right]} \\
C = \frac{\pi \beta B_e}{4 p B_{fer}} D \\
p = \frac{\pi D}{\Delta_p}
\end{cases}
\end{aligned}
\tag{2.24}
$$

Le modèle compte 17 paramètres dont 7 fixés. Par conséquent, 10 paramètres sont à déterminer. Le modèle contient 6 contraintes d'égalité indépendantes, ce qui réduit le nombre de degrés de liberté à 4. Le reste des paramètres sont des sorties.

La reformulation consiste à :

- Reformuler les contraintes de telle sorte à s'assurer que les sorties soient exclusivement des paramètres variables.
- Modifier l'ordre des équations afin de pouvoir calculer les sorties en fonction des entrées.

Le modèle reformulé ainsi obtenu est le suivant :

$$
\begin{aligned}
&min\, V_u = \pi \frac{D}{\lambda}(D + E - e - l_a)(2C + E + e + l_a) \\
&s.c \begin{cases}
D = \frac{p \Delta_p}{\pi} \\
J_{cu} = \sqrt{\frac{E_{ch}}{k_r E}} \\
B_e = \frac{2 l_a M}{D log\left[\frac{D+2E}{D-2(l_a+e)}\right]} \\
K_f = 1.5 p \beta \frac{e+E}{D} \\
\lambda = \frac{\pi}{2\Gamma_{em}}\left(1 - K_f\right)\sqrt{k_r \beta E_{ch} E} D^2 (D + E) B_e \\
C = \frac{\pi \beta B_e}{4 p B_{fer}} D
\end{cases}
\end{aligned}
\tag{2.25}
$$

Deux configurations ont été comparées : l'optimisation avec et sans la reformulation. Les tests ont été réalisés avec une précision sur la largeur des boites $\epsilon = 10^{-7}$. Les principaux résultats

sont recensés dans le Tableau 2-5 et le Tableau 2-6. Dans ces tableaux, la première et la dernière mise à jour de la borne supérieure ainsi que quelques mises à jour intermédiaires de cette borne sont présentées avec à chaque fois la durée nécessaire et le nombre d'itérations correspondants.

Sans la reformulation, la première borne supérieure a été trouvée après 14142 itérations et la meilleure solution trouvée au bout de 19 millions d'itérations (voir Tableau 2-5). C'est une valeur très élevée sachant que les contraintes d'égalité ont été relâchées avec une tolérance $\epsilon_t = 10^{-4}$ afin de rendre possible l'étape de mise à jour de la borne. Ainsi, une contrainte $p_i = h(P)$ est transformée en $p_i \in [h(P) - \epsilon_t, h(P) + \epsilon_t]$. Pour des modèles plus lourds en calcul, cette méthode devient donc impossible à réaliser.

Tableau 2-5. Sans la reformulation

Borne supérieure (1.e-4)	Durée (s)	Itérations
9.16	1.8	14142
7.21	13	105241
6.44	34	270232
6.12	221	1974459
6.07	2942	19716202

Tableau 2-6. Avec la reformulation

Borne supérieure (1.e-4)	Durée (s)	Itérations
10.76	0.04	4
8.55	0.076	17
7.15	0.093	68
7.07	0.125	158
6.67	0.327	1486
6.57	0.343	1595
6.52	0.359	1703
6.32	0.421	2220
6.15	0.483	2705
6.084	1.263	9077
6.083	3.199	25136
6.078	17.209	125851
6.073	18.894	133282

Avec la reformulation, nous avons constaté une nette amélioration au niveau du nombre d'itération expliqué par la facilité de mettre à jour la borne supérieure (voir Tableau 2-6). En effet, la première mise à jour a été faite au bout de 4 itérations seulement et la meilleure solution trouvée au bout de 133282 itérations. La Figure 2-7 montre que la mise à jour de la borne supérieure devient beaucoup plus facile. Elle permet d'atteindre des valeurs très proches de l'optimum global seulement après 9000 itérations et une seconde de calcul (6.08 e-4). Sachant qu'aucune relaxation des contraintes n'a été réalisée, la reformulation a conduit à

une nette amélioration de la convergence de l'algorithme ainsi que la qualité de l'optimum trouvé. En effet, le nombre d'itérations ainsi que la durée de l'optimisation enregistrent un gain de 99% avec la reformulation (Tableau 2-7).

Tableau 2-7. Optimisation du moteur avec et sans reformulation

	Moteur	
	Itérations (#)	Durée (s)
Sans reformulation	19716202	2942
Avec reformulation	133282	18.894
Gain	99%	99%

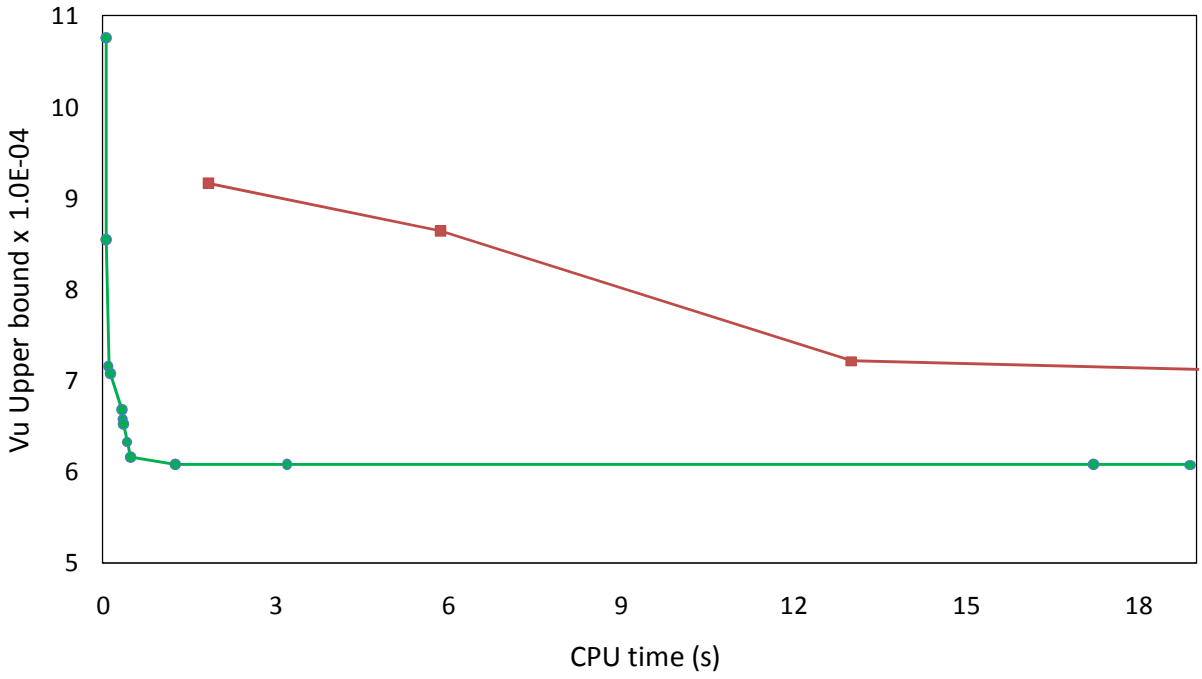

Figure 2-7. Evolution de la borne supérieure avec (vert) et sans (rouge) reformulation

2.4. Conclusion

Dans cette partie, un algorithme déterministe d'optimisation globale a été présenté. Il est basé sur le calcul d'intervalles. Cet algorithme est valable pour les modèles d'optimisation avec équations algébriques et sans fonctionnelles. Il nécessite la reformulation du modèle afin de permettre de faire l'évaluation par intervalles via les fonctions d'inclusion. La contribution dans ce chapitre consiste en une reformulation qui vise à accélérer la convergence de l'algorithme IBBA. Cette reformulation consiste en la réduction du nombre de variables du problème pour traiter uniquement les variables de décision. Elle consiste aussi en la reformulation des équations et leur réorganisation afin de permettre le calcul des sorties via les entrées d'une part, et de faciliter le respect des contraintes d'autre part. Les tests numériques ont montré l'apport de cette reformulation en termes de nombre d'itérations et de précision de l'optimum obtenu.

Cette contribution a fait l'objet de deux papiers respectivement dans le journal Engineering Application of Artificial Intelligence [Mazhoud I. et al, 2012a] et IEEE transaction on Magnetics [Mazhoud I. et al, 2012b], d'une participation à la conférence internationale IEEE Compumag [Mazhoud I. et al, 2011a] et à la conférence nationale ROADEF [Mazhoud I. et al, 2011b].

Cette méthode a donc des avantages indéniables concernant la garantie. En revanche, elle a ses limites. En effet, c'est une méthode déterministe globale qui permet de garantir l'obtention de l'optimum s'il existe et de prouver sa non-existence en cas de non convergence. Cette garantie a un prix : la complexité. En réalité, et même avec la reformulation proposée, cette méthode reste applicable pour une faible partie des problématiques réelles et se limite dans la pratique aux problèmes avec 4 à 5 degrés de liberté. C'est de là que le prochain chapitre puise son intérêt. Dans le prochain chapitre, une méthode d'optimisation qui traitera les problèmes de plus grande dimension sera introduite.

3. Optimisation stochastique : PSO avec nouveau mécanisme de gestion de contraintes

Dans cette partie, une méthode d'optimisation globale stochastique est présentée. Ces méthodes tirent leur intérêt de leur vaste champ d'application. En effet, elles permettent de résoudre des problèmes d'optimisation de grande dimension sous forme de boite noire ou blanche. Aucune information sur les équations ni sur les dérivées n'est nécessaire. Ceci leur permet d'avoir un champ d'application très large et une mise en pratique très facile. Elles peuvent s'appliquer aussi bien aux modèles analytiques qu'aux modèles numériques très fréquemment rencontrés en pré-dimensionnement de produits (logiciels de simulations, modèles éléments finis, codes de calcul, …).

L'algorithme décrit dans cette partie est un algorithme d'optimisation par essaim particulaire ou PSO (Particle Swarm Optimization). L'algorithme PSO est inspiré du comportement social des groupes d'animaux comme les oiseaux ou les poissons à la recherche de sources de nourriture. Ces groupes utilisent des capacités cognitives et sociales particulières afin de trouver rapidement les endroits abondants en nourriture.

Deux contributions seront présentées dans cette partie. La première contribution consiste en un nouveau mécanisme de gestion de contraintes. Ce mécanisme a été testé et comparé à d'autres mécanismes existants sur un benchmark de problèmes d'optimisation. Il a également été testé sur le problème de conception du moteur électrique à aimants permanents présenté dans le paragraphe 1.2.

La deuxième contribution consiste en l'extension du PSO aux modèles dynamiques. Cette extension combine le PSO avec le nouveau mécanisme de gestion de contraintes avec une méthode numérique de résolution d'équations différentielles ordinaires (ODE). Cet algorithme sera testé sur l'optimisation de l'actionneur différentiel présenté dans le paragraphe 1.2.2.

3.1. PSO : algorithme de base pour l'optimisation sans contraintes

PSO est un algorithme stochastique principalement dédié aux problèmes d'optimisation continus. C'est une méta-heuristique d'optimisation introduite en 1995 par Kennedy et Eberhart [Kennedy, Eberhart, 1995] et s'inspire de l'observation du comportement social de populations d'animaux. Cette méthode a prouvé son efficacité pour résoudre des problèmes d'ingénierie et a reçu un intérêt grandissant des communautés de l'optimisation et de l'ingénierie. En plus de son efficacité, PSO se caractérise par une facilité d'implémentation et un nombre réduit de paramètres de réglage.

PSO a été initialement proposé pour résoudre des problèmes d'optimisation non-contraints. Il est initialisé avec un ensemble de particules. Chaque particule correspond à une solution candidate. Cette population de particules est mise à jour à chaque itération (appelée aussi génération). PSO trouve l'optimum en déplaçant les particules dans l'espace de recherche avec des vitesses pondérées aléatoirement.

Prenons l'exemple d'un problème de minimisation sans contraintes (n)-dimensionnel :

$$\min_{x \in X \subseteq \mathbb{R}^n} f(x) \tag{3.1}$$

avec $X = [l_i, u_i]$ pour $i = 1 \dots n$ le domaine de recherche. Soit m $(j = 1 \dots m)$ la taille de l'essaim c'est-à-dire le nombre de particules. La position actuelle de la $j^{ème}$ particule à chaque itération t de l'algorithme est donnée par un vecteur (n)-dimensionnel $x^j(t) = \left[x_i^j(t), \right]^T$, pour $i = 1, \dots, n$. Sa vitesse actuelle (à l'itération en cours) est $v^j(t) = \left[v_i^j(t), \right]^T$ pour i=1, … , n. La fitness de la particule correspond à son évaluation par la fonction objectif et est donnée par $y^j(t) = f\left(x^j(t)\right)$. La nouvelle vitesse de la particule $v^j(t+1)$ est calculée en fonction de sa meilleure position atteinte jusque là $xp^j(t)$ et de la meilleure position atteinte par tout l'essaim $xg^j(t)$ suivant cette formule [Shi, Eberhart, 1998a] :

$$v^j(t+1) = w.v^j(t) + c_1.r_1.(xp^j(t) - x^j(t)) + c_2.r_2.(xg^j(t) - x^j(t)) \tag{3.2}$$

avec w le facteur d'inertie introduit pour contrôler les variations excessives des positions d'une particule en dehors de l'espace de recherche. c_1 et c_2 sont deux paramètres d'accélération appelés respectivement paramètre cognitif et paramètre social. r_1 et r_2 sont deux vecteurs aléatoires suivants la distribtion uniforme $\mathcal{U}[0,1]$ introduits pour maintenir la diversité de l'essaim.

La position $x^j(t+1)$ de la $j^{ème}$ particule à l'itération $t+1$ est mise à jour en ajoutant la nouvelle vitesse à la position précédente :

$$x^j(t+1)=x^j(t)+v^j(t+1) \tag{3.3}$$

La meilleure position d'une particule et la meilleure position de tout l'essaim sont mise à jour à chaque itération :

$$\begin{cases} xp^j(t) = \underset{h\in\{xp^j(t-1),x^j(t)\}}{argmin} f(h) \\ yp^j(t) = f\left(xp^j(t)\right) \end{cases} \quad \forall t, \forall j=1,\dots,m \tag{3.4}$$

$$\begin{cases} xg(t) = \underset{h\in\{xp^1(t)\dots xp^m(t)\}}{argmin} f(h) \\ yg(t) = f\big(xg(t)\big) \end{cases} \quad \forall t \tag{3.5}$$

L'algorithme s'assure que les particules restent dans le domaine de recherche défini. Pour cela, les contraintes de domaines sont forcées en considérant pour chaque variable x_i la valeur de la borne la plus proche comme suit [Vaz, Vincente, 2007] :

$$x_i^j(t) = \begin{cases} l_i & si\ x_i^j(t) < l_i \\ u_i & si\ x_i^j(t) > u_i \\ x_i^j(t) & sinon \end{cases} \tag{3.6}$$

Le processus est répété jusqu'à atteindre un critère d'arrêt défini : nombre d'itérations maximal, vitesse proche de 0, durée maximale ...

3.2. PSO : mécanismes existants de gestion de contraintes

Dans cette partie, on utilise la reformulation introduite dans le chapitre sur l'optimisation par intervalles. Ainsi, on part directement sur le problème contraint reformulé :

$$\begin{cases} \min\limits_{x \in X \subseteq \mathbb{R}^n} f(x) \\ y = g(x) \\ x \in [\underline{x}, \overline{x}], y \in [\underline{y}, \overline{y}] \end{cases}$$

Etant donné leur aspect aléatoire, les contraintes d'égalité ne sont pas adaptées aux algorithmes stochastiques. En effet, il est statistiquement impossible de satisfaire une contrainte d'égalité en choisissant aléatoirement un point dans l'espace de recherche en l'absence d'informations sur l'équation. C'est principalement pour cette raison que la reformulation est utilisée. Dans le cas où $\underline{y} = \overline{y}$, le domaine est relâché avec une précision ϵ et devient $[\underline{y} - \epsilon, \overline{y} + \epsilon]$.

Pour traiter les contraintes, divers mécanismes de gestion de contraintes ont été proposés. La revue de littérature montre que les fonctions de pénalité et les règles basées sur la faisabilité ont été les mécanismes les plus implémentés pour traiter les contraintes. Parsopoulos et Vrahatis [Parsopoulos, Vrahatis, 2005] ont proposé une fonction de pénalité à affectation multi-stage non-stationnaire. Ils utilisent les facteurs de pénalité pour calculer la somme des violations des contraintes. Sedlaczek et Eberhart [Sedlaczek, Eberhart, 2006] ont aussi présenté une extension de la fonction de pénalité non-stationnaire, à savoir le multiplicateur de Lagrange augmenté. He et Wang [He, Wang, 2007a] ont implémenté un PSO co-évolutionnaire dans lequel le PSO est exécuté avec deux types d'essaims. Ces deux essaims évoluent de façon interactive dans l'espace des solutions et l'espace des facteurs de pénalité. Wang et Yin [Wang, Yin, 2008] ont proposé un nouveau critère de classement des particules afin de les guider vers les zones faisables. La fonction objectif et les contraintes sont traitées séparément et les particules faisables sont ordonnées suivant leur fitness. Plusieurs autres méthodes ont été implémentées comme des extensions des algorithmes évolutionnaires multi-objectifs. Par exemple la méthode IS-PAES basée sur la stratégie évolutionnaire (ES) Pareto archivée [Aguirre et al, 2004]. Wang et al. ont proposé un modèle à compromis adaptatif (ATM) pour les problèmes d'optimisation sous contraintes [Wang et al, 2008]. Dans un papier plus récent, Wang et al. ont proposé une nouvelle approche nommée AATM utilisant une technique de rétrécissement d'espace qui vise à accélérer ATM. La technique de rétrécissement d'espace est proposée par Aguirre et al. [Aguirre et al, 2004]. Dans cette technique, l'effort de recherche est concentré dans une zone spécifique de l'espace faisable en rétrécissant l'espace contraint. Hu et Eberhart ont proposé une technique de préservation de faisabilité pour traiter les contraintes [Hu, Eberhart, 2002]. Dans cette technique, PSO est initialisé avec un ensemble de particules faisables et une fonction de faisabilité est utilisée

pour vérifier si les nouvelles positions respectent les contraintes. D'autres méthodes basées sur la préservation de faisabilité ont été développées par He et al. [He et al , 2004] dans lesquelles chaque particule suit uniquement des positions faisables. Aguirre et al. ont introduit une technique basée sur la faisabilité et la somme des violations des contraintes en utilisant un fichier externe pour stocker les particules tolérables [Aguirre et al, 2004]. Pulido et Coello [Pulido, Coello, 2004] ont présenté un mécanisme simple de gestion de contraintes basé sur l'éloignement des particules de la région faisable pour choisir une particule leader. De plus, l'algorithme incorpore un opérateur de turbulence qui améliore les capacités exploratoires du PSO. Lu et Chen [Lu, Chen, 2006] et He et Wang [He, Wang, 2007b] ont aussi appliqué d'autres méthodes basées sur la règle de faisabilité. Dans la même idée, Worasucheep [Worasucheep, 2008] a proposé d'utiliser un mécanisme basé sur la règle de faisabilité combiné avec des mécanismes de détection de stagnation et de dispersion qui peuvent détecter une éventuelle stagnation des particules et les disperser dans ce cas. Lu et Chen [Lu, chen, 2008] ont présenté un PSO modifié, appelé PSO à vitesse auto-adaptative. Dans cet algorithme, la gestion des contraintes se fait par le biais d'une gestion de contraintes par fonction objectif dynamique (dynamic-objective constraint-handling method). Montes et Coello [Montes, Coello, 2005] ont utilisé un mécanisme de diversification appelé SMES pour garder des particules qui violent peu les contraintes dans la prochaine population. Liu et al. [Liu et al, 2010] ont proposé un PSO hybride nommé PSO-DE entre PSO et un algorithme d'évolution différentielle (DE) pour résoudre des problèmes numériques contraints et des problèmes d'ingénierie. L'algorithme d'évolution différentielle est utilisé pour mettre à jour les meilleures positions des particules afin de les forcer à éviter les stagnations.

3.3. Contribution : nouveau mécanisme de gestion de contraintes

3.3.1. Mécanisme de gestion de contraintes

Souvent, les problèmes d'ingénierie sont contraints par le cahier des charges et il est parfois difficile de trouver « manuellement » une solution qui respecte les contraintes sans l'aide d'algorithmes adaptés. Le mécanisme de gestion de contraintes proposé est basé sur une évaluation de la distance de chaque particule à la région faisable. Ce mécanisme « pousse » les particules dans la région faisable si elles sont localisées en dehors. Une fois ceux-ci dans la région faisable, le mécanisme améliore leurs fitness afin de trouver l'optimum global. Ce mécanisme affecte directement le mécanisme de sélection de la meilleure position de chaque particule et la meilleure position de tout l'essaim. En effet, l'idée est de favoriser les particules les plus proches de la région faisable même si au niveau de leur fitness elles ne sont pas performantes.

Reprenons le problème d'optimisation contraint sous la forme suivante :

$$\begin{cases} \min\limits_{x \in X \subseteq \mathbb{R}^n} f(x) \\ y = g(x) \\ x \in [\underline{x}, \overline{x}], y \in [\underline{y}, \overline{y}] \end{cases}$$

où $y = \{y_1 \dots y_m\}$ et $g = \{g_1 \dots g_m\}$ (m à ne pas confondre avec le nombre de particules). A chaque itération, la fonction de violation totale des contraintes est utilisée pour évaluer la somme des violations de chaque particule. Cette fonction évalue la distance de chaque position à la région faisable. On propose d'utiliser l'arithmétique d'intervalles pour normaliser les violations afin de calculer une violation totale. Par conséquent, pour chaque particule, deux informations seront disponibles : la fitness et la violation totale. Une fois ces deux informations calculées, une méthode de classement lexicographique est utilisée pour comparer les particules. En effet, les deux critères de classement (fitness et violation totale) ont un ordre de traitement : la violation totale est préférée à la fitness. Ceci implique qu'une particule avec une moindre violation et une fitness moins bonne est systématiquement préférée à une particule qui est plus performante en termes de fonction objectif mais qui viole plus les contraintes.

Etant donné un point x faisable, une contrainte d'inégalité $g_k \leq 0$ est dite active en x si $g_k(x) = 0$ et inactive (passive) si $g_k(x) < 0$. Une contrainte d'égalité $h_k = 0$ est active en tout point faisable ($h_k(x) = 0$) [Nocedal, Wright, 2006].

Une règle basée sur la faisabilité est introduite pour évaluer la violation d'une contrainte afin de guider les particules vers les régions faisables. Soit $x^j(t)$ la position de la particule j à l'itération t. Les contraintes $g_k(x)$ sont calculées et les violations évaluées en considérant la distance d de la particule par rapport à la région faisable. En effet :

$$d\left(y_k^j(t)\right) = d\left(g_k\left(x^j(t)\right)\right) \leftarrow \begin{cases} \underline{y_k} - y_k^j(t)\ Si\ y_k^j(t) < \underline{y_k} \\ y_k^j(t) - \overline{y_k}\ Si\ y_k^j(t) > \overline{y_k} \\ 0 \qquad\qquad Sinon \end{cases} \tag{3.7}$$

<u>Propriété 1</u>: La violation d'une contrainte est toujours supérieure ou égale à 0.

Les contraintes peuvent avoir des ordres de magnitude très différents. Par conséquent, une ou plusieurs contraintes peuvent dominer et masquer l'effet des autres contraintes. Pour cela, il est nécessaire d'effectuer une normalisation avant de calculer la violation totale afin de rendre les contraintes comparables. Dans l'approche que nous proposons, la violation de chaque contrainte est normalisée par rapport à la violation maximale dans le domaine de recherche.

Soit dp_k^{max}la plus grande violation possible de la contrainte g_k. Afin d'évaluer dp_k^{max}, on considère la boite $X = [l_i, u_i], i = 1 \dots n$. Pour chaque contrainte g_k:

$$Image(g_k, X) = \{g_k(x) \,/\, x \in X\} := [\min_{x \in X} g_k(x)\,, \max_{x \in X} g_k(x)] \tag{3.8}$$

Pour estimer l'image d'une contrainte sur une boite, on propose d'utiliser l'arithmétique d'intervalles présentée dans la section précédente. Soient G_k la fonction d'inclusion de g_k:

$$G_k(X) = [LB_k, UB_k] \tag{3.9}$$

Ainsi, la plus grande violation d'une contrainte est calculée suivant 5 cas (Figure 3-1):

$$dp_k^{max} = \begin{cases} UB_k - \overline{y_k} & Si\ (UB_k \geq \overline{y_k}) \wedge \left(LB_k \geq \underline{y_k}\right) \\ \underline{y_k} - LB_k & Si\ \left(LB_k \leq \underline{y_k}\right) \wedge (UB_k \leq \overline{y_k}) \\ max\left(UB_k - \overline{y_k}, \underline{y_k} - LB_k\right) & Si\ \left[\underline{y_k}, \overline{y_k}\right] \subseteq [LB_k, UB_k] \\ 0 & Si\ [LB_k, UB_k] \subseteq \left[\underline{y_k}, \overline{y_k}\right] \\ NaN & Si\ [LB_k, UB_k] \cap \left[\underline{y_k}, \overline{y_k}\right] = \emptyset \end{cases} \tag{3.10}$$

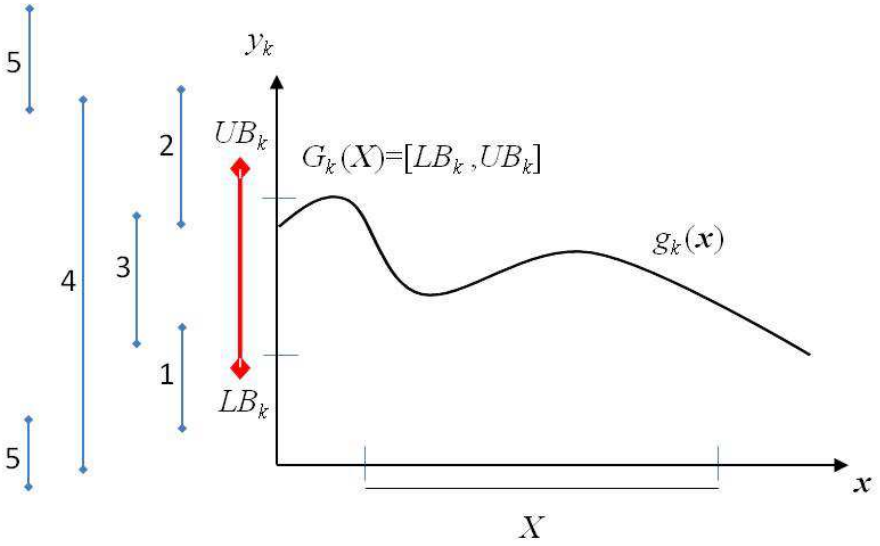

Figure 3-1. Evaluation de la violation : les différents cas

Exemple : Soit la contrainte $g_1(x) = x(x-1)+1 \leq 0$ et $X = [0\,;2]$. L'extension naturelle G_1 de $g_1$donne $G_1(X) = [LB_1, UB_1] = [-1{,}3]$. Par consequent, la violation maximale de g_1 est $UB_1 = 3$.

<u>Propriété 2</u> : La violation maximale d'une contrainte est supérieure ou égale à 0 : $\forall k, dp_k^{max} \geq 0$. *Si* $\exists k$ *tel que* $dp_k^{max} = NaN$ *alors le problème n'a pas de solutions.*

Remarque : Le cas $[LB_k, UB_k] \cap \left[\underline{y_k}, \overline{y_k}\right] = \emptyset$ correspond en réalité à une contrainte non realisable. C'est-à-dire que pour aucun point dans X la contrainte ne peut être satisfaite. Dans ce cas, il faut revoir la contrainte g_k et/ou le domaine de recherche X.

Pour toute contrainte, la violation d'une particule est normalisée comme suit :

$$dp_k\left(x^j(t)\right) = \frac{d\left(g_k\left(x^j(t)\right)\right)}{dp_k^{max}} \; si \; d\left(g_k\left(x^j(t)\right)\right) \neq 0 \; \forall k, \forall j, \forall t \tag{3.11}$$

La violation totale d'une particule est définie comme la somme des violations normalisées de toutes les contraintes :

$$dp\left(x^j(t)\right) = \sum_{k=1}^{p} dp_k\left(x^j(t)\right) \; \forall j, \forall t \tag{3.12}$$

Lors de la mise à jour des particules leaders (meilleure position d'une particule et meilleure position de l'essaim), notre approche consiste à choisir la particule qui est plus proche de la région faisable. Ainsi, on considère la minimisation de la violation totale $dp(x)$ plus

importante que la minimisation de la fonction objectif $f(x)$. La particule qui minimise $dp(x)$ est toujours préférée. Si deux particules ont la même violation totale $dp(x)$ alors la particule qui a la meilleure fitness est préférée.

Par conséquent, les problèmes d'optimisation P_1 et P_2 suivants sont résolus de façon séquentielle :

$$min_{x\in X}[f(x), dp(x)] \leftarrow \begin{cases} P_1: \quad dp^* \leftarrow \min\limits_{x\in X} dp(x) \\ P_2: \min\limits_{x\in X/dp(x)=dp^*} f(x) \end{cases} \tag{3.13}$$

Mise à jour de la meilleure position de chaque particule

A l'itération t, la nouvelle position de la particule j est comparée à sa meilleure position $xp^j(t-1)$. La nouvelle meilleure position est déterminée en résolvant séquentiellement les deux problèmes P_1 et P_2 suivants :

$$xp^j(t) \leftarrow \begin{cases} P_1: A = \underset{h\in\{xp^j(t-1), x^j(t)\}}{argmin} dp(h) \\ P_2: B = \underset{h\in A}{argmin}\, f(h) \end{cases} \quad \forall j, \forall t \tag{3.14}$$

Ici, deux cas de figure sont à distinguer [2]:

- Si $\|A\| = 1$, ce qui signifie que soit la nouvelle position ou l'ancienne meilleure position a la violation minimale (ou que les deux violations sont différentes). Ici inutile de résoudre P_2.
- Si $\|A\| = 2$, dans ce cas la nouvelle position et l'ancienne meilleure position ont la même violation totale, donc il est nécessaire de déterminer la meilleure position des deux en se basant sur la fonction objectif comme critère.

Mise à jour de la meilleure position de tout l'essaim

A l'itération t, les meilleures positions $xp^j(t)$ mise à jour par la procédure expliquée dans le paragraphe précédent sont comparées entre elles afin de mettre à jour la meilleure position de tout l'essaim $xg(t)$. La meilleure position de l'essaim à l'itération t est mise à jour en résolvant séquentiellement les deux problèmes P_1 et P_2 suivants :

$$xg(t) \leftarrow \begin{cases} P_1: A = \underset{h\in\{xp^1(t)\ldots xp^m(t)\}}{argmin} dp(h) \\ P_2: B = \underset{h\in A}{argmin}\, f(h) \end{cases} \quad \forall t \tag{3.15}$$

[2] $\|A\|$: cardinalité de l'ensemble A

Deux cas sont à distinguer ici :

- Si $\|A\| = 1$, ce qui signifie qu'une seule particule à la violation la plus faible. Ici inutile de résoudre P_2 car c'est cette particule qui sera la nouvelle meilleure position de l'essaim.
- Si $\|A\| \geq 2$, dans ce cas au moins deux particules ont la meilleure violation. Dans ce cas, il est nécessaire de déterminer la meilleure position en se basant sur la fonction objectif comme critère.

Algorithme 3-1. PSO avec le mécanisme de gestion de contraintes proposé

1: Choose a swarm size m, a stopping criterion and PSO parameters (c_1& c_2)

2: Initialization ($t = 0$)

For j from 1 to m

3: randomly generate an initial particle position $x^j(0)$ and velocity $v^j(0)$

4: compute the constraints values $g_k(\, x^j(0))$

5: compute the total constraints violation $dp(\, x^j(0))$

6: update the particle's best position : $xp^j(0) := x^j(0)$

End For

7: update the initial global best position $xg(0)$

Repeat

8: $t := t + 1$

For j from 1 to m

9: update particle's velocity $v^j(t)$ (Equ. 3.2) and position $x^j(t)$ (Equ. 3.3)

10: enforce the bound constraints (Equ. 3.6)

11: compute the constraints values $g_k(\, x^j(t))$

12: compute the total constraints violation $dp(\, x^j(t))$

13: update the particle's best position $xp^j(t)$

End For

14: update the global best position $xg(t)$

Until stopping criterion is met

3.3.2. Tests numériques

L'algorithme PSO proposé a été implémenté et testé en Java et Matlab version 7.8. Les tests ont été accomplis sur une station de travail avec un processeur Intel Core™ 2 cadencé à 2.66 Ghz avec 8 Gb de ram. Pour l'arithmétique d'intervalle c'est la très répandue bibliothèque IntLab qui a été utilisée sur Matlab [Rump S. M., 1999].

Dans toutes les expérimentations une population $m = 50$ particules a été utilisée car ceci a démontré des performances suffisantes pour résoudre plusieurs problèmes d'ingénierie. Le critère d'arrêt stoppe l'exécution de l'algorithme au bout d'un nombre d'itérations $Iter = 200$. Ceci correspond à 10000 évaluations de la fonction objectif (FFE : Fitness Function Evaluation). 20 tests ont été performés pour chaque problème afin d'évaluer la stabilité de l'algorithme. Le facteur d'inertie w peut varier au cours des iterations [Shi Y., Eberhart R., 1998B]. Comme suggéré dans [Janson S. et al, 2008], nous avons choisi un facteur d'inertie décroissant avec les itérations afin de mieux équilibrer l'exploration (recherche globale) et l'intensification (recherche locale) et permettre ainsi une meilleure convergence :

$$w = 0.5 + \frac{1}{2(ln(t) + 1)} \tag{3.16}$$

[Arumugam M. et al, 2008] suggèrent de choisir les paramètres c_1 et c_2 comme fonction des meilleures valeurs locales et globales de la fonction objectif. D'autres études expérimentales suggèrent d'utiliser une relation empirique entre w, c_1 et c_2 [Van Den Bergh F., 2001]. Plusieurs expérimentations ont été effectuées sur l'algorithme proposé et il est apparu que choisir c_1 et c_2 proches de 2 permet d'avoir les meilleures performances dans tout le benchmark.

Dans l'objectif d'évaluer les performances de l'algorithme proposé que nous appelons CVI-PSO pour «Constraint Violation by Interval analysis PSO », plusieurs tests ont été effectués sur un benchmark. Ce benchmark contient 24 problèmes mathématiques et 3 problèmes d'ingénierie décrits dans [Wang Y. et al, 2009], [Aragon V. S. et al, 2010] et [He Q., Wang L., 2007a]. Ces problèmes d'optimisation ont été testés avec plusieurs algorithmes. Les résultats obtenus par CVI-PSO seront comparés à ceux des autres approches.

Le benchmark inclue plusieurs fonction objectifs (linéaire, non linéaire, polynomiale, quadratique…) et contraintes (égalité, inégalité, linéaire, non linéaire…). Les caractéristiques du benchmark sont reportées dans le Tableau 3-1 avec Var# le nombre de variables, Obj-f le type de la fonction objectif, LInC# le nombre de contraintes d'inégalité linéaires, NLInC# le nombre de contraintes d'inégalité non-linéaires, LEC# le nombre de contraintes d'égalité linéaires, et NLEC# le nombre de contraintes d'égalité non linéaires. Le coefficient de faisabilité α (%) est défini comme étant le pourcentage de solutions faisables parmi 10^6 solutions générées aléatoirement. Le coefficient α défini une mesure de la difficulté du

problème à résoudre et ses valeurs sont très proches de celles reportées dans [Wang et al, 2009].

L'initialisation de l'essaim a un impact très important sur la qualité de la convergence du PSO. [Campana et al, 2010] ont donné des indications théoriques et numériques sur le choix de la population initiale. Dans nos expérimentations, on a intégré deux types d'initialisation. La première façon est complètement aléatoire où les positions sont choisies aléatoirement avec une fonction de répartition uniforme dans le domaine de recherche initial. La deuxième façon d'initialiser l'essaim se base sur la méthode du Latin-Hypercube qui divise l'espace de recherche afin de choisir des positions qui le balayent au mieux. Dans les tests du benchmark, nous utiliserons l'instanciation aléatoire afin d'évaluer l'algorithme en lui-même. Dans le test numérique sur le moteur à aimants permanents, nous utiliserons les deux types d'instanciation afin de voir les différences sur la qualité de la convergence.

Pour le détail des équations des modèles, se référer à l'annexe 1.

Tableau 3-1. Caractéristiques du benchmark : problèmes mathématiques

Pb	Var#	Obj-f	LInC#	NLInC#	LEC#	NLEC#	$\alpha(\%)$
g01	13	Quadratic	9	0	0	0	0.0001
g02	20	Non-linear	1	1	0	0	99.99
g03	10	Polynomial	0	0	0	1	0.000
g04	5	Quadratic	0	6	0	0	26.92
g05	4	Cubic	2	0	0	3	0.000
g06	2	Cubic	0	2	0	0	0.0054
g07	10	Quadratic	3	5	0	0	0.0002
g08	2	Non-linear	0	2	0	0	0.86
g09	7	Polynomial	0	4	0	0	0.53
g10	8	Linear	3	3	0	0	0.0002
g11	2	Quadratic	0	0	0	1	0.00
g12	3	Quadratic	0	1	0	0	4.54
g13	5	Non-linear	0	0	0	3	0.00
g14	10	Non-linear	0	0	3	0	0.0
g15	3	Quadratic	0	0	1	1	0.00
g16	5	Non-linear	4	34	0	0	0.018
g17	6	Non-linear	0	0	0	4	0.00
g18	9	Quadratic	0	13	0	0	0.00
g19	15	Non-linear	0	5	0	0	35.54
g20	24	Linear	0	6	2	12	0.00
g21	7	Linear	0	1	0	5	0.00
g22	22	Linear	0	1	8	11	0.00
g23	9	Linear	0	2	3	1	0.00
g24	2	Linear	0	2	0	0	74.26

Tableau 3-2. Caractéristiques du benchmark : problèmes d'ingénierie

Pb	Var#	Obj-f	LInC#	NLInC#	LEC#	NLEC#	$\alpha(\%)$
eng01	4	Non-linear	3	1	0	0	39.7
eng02	4	Non-linear	2	5	0	0	2.66
eng03	3	Non-linear	1	3	0	1	0.75

3.3.2.1. **Les 24 problèmes mathématiques**

Les problèmes mathématiques du benchmark ont été utilisés dans les tests numériques de plusieurs algorithmes d'optimisation. Ils ont été résolus auparavant en utilisant les algorithmes suivants : un modèle d'accélération adaptatif utilisant une technique de rétrécissement d'espace appelée AATM [Wang et al, 2009], une heuristique inspirée par le modèle T-Cell du système immunitaire [Aragon et al, 2010] et des variantes combinées de l'algorithme d'évolution différentielle appelé DECV [Mezura et al, 2010].

Les résultats d'optimisation par les algorithmes mentionnés plus haut ainsi que par CVI-PSO sont présentés dans le Tableau 3-3. Ces résultats indiquent les meilleures solutions obtenues, les pires, les moyennes et les écart-types (parmi les 20 exécutions). Avec l'algorithme CVI-PSO, aucune relaxation n'a été faite au niveau des contraintes d'égalité et dans les solutions obtenues, les contraintes d'égalité sont toujours actives (c'est-à-dire respectées). Ceci n'est pas le cas des autres algorithmes.

La Figure 3-2 montre l'évolution de la fonction objectif et de la violation des contraintes du problèmes g05. On voit qu'au début du processus d'optimisation, la fonction objectif peut être détérioré mais toujours au profit d'une solution qui améliore la faisabilité. Une fois la violation nulle (environ itération 50), la fonction objectif devient prioritaire.

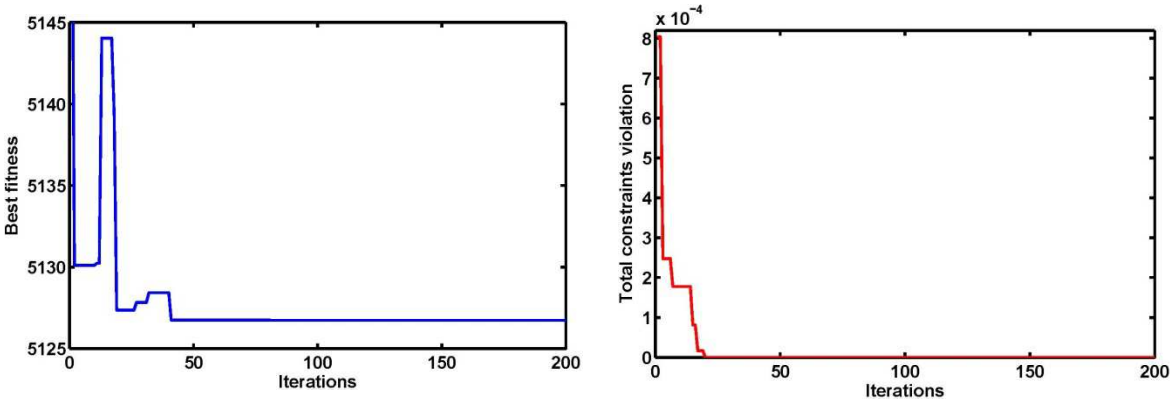

Figure 3-2. Evolution de la Fitness et de la violation pour le problème g05

Les résultats numériques montrent que CVI-PSO réalise de bons résultats pour tous les problèmes. Comme décrit dans le Tableau 3-3, CVI-PSO arrive à trouver les optima globaux des problèmes g01, g03, g06, g11, g12, g16, g17 et g24 et trouve des solutions très proches de l'optimum global pour le reste des problèmes.

Il est aussi intéressant de noter que pour les problèmes g01, g03, g11 et g12 les meilleures solutions obtenues correspondent aux optima globaux théoriques avec des écart-types très proches de 0 pour g01 et g03 et égales à 0 pour g11 et g12.

Pour le problème g20, CVI-PSO trouve des solutions faisables en seulement une exécution parmi les 20 alors que les autres algorithmes n'ont pas réussi. Pour le problème g22, tous les algorithmes n'ont pas réussi à trouver des solutions faisables dans aucune exécution. Ces

résultats peuvent s'expliquer par la difficulté de ces problèmes avec $\alpha = 0$, de plus grande dimensionnalité (resp. 24 et 22 variables), plus de contraintes d'égalité (resp. 14 et 19) et plus de contraintes non linéaires (resp. 18 et 12).

Pour les problèmes g05, g17 et g23, CVI-PSO trouve toujours les mêmes meilleures solutions avec des écart-types très proches à 0. Il est important de noter que même si CVI-PSO trouve toujours les mêmes solutions, les meilleures solutions trouvées dans d'autres travaux sont meilleures. Les solutions trouvées par le CVI-PSO rendent actives toutes les contraintes d'égalité alors que dans les autres méthodes ces contraintes ont été relâchées.

Pour le problème g24, malgré ses régions faisables disjointes, le CVI-PSO n'a aucune difficulté à converger toujours vers la meilleure solution connue avec une écart-type très proche de 0. Ce problème avec une faible dimensionnalité (deux variables) a une fonction objectif linéaire et seulement deux contraintes d'inégalité.

Les résultats numériques sur le benchmark des problèmes mathématiques montrent que CVI-PSO est un bon algorithme qui n'éprouve pas beaucoup de difficultés à converger vers les meilleures solutions avec des écart-types souvent faibles. Il fait donc preuve de stabilité dans sa convergence. Toutefois, les résultats démontrent que les problèmes avec une grande dimensionnalité et avec un grand nombre de contraintes d'inégalité sont difficiles à résoudre. Selon notre expérience, ceci peut être amélioré en augmentant le nombre de particules de l'essaim.

Pour les problèmes avec des contraintes d'égalité, le CVI-PSO a trouvé des optima équivalents ou même meilleures que les meilleurs résultats connus sans utiliser des relaxations.

Tableau 3-3. Résultats sur le benchmark : problèmes mathématiques

Pb	Stat	CVI-PSO	AATM	DECV	T-CELL
g01	Best	-14.99	-15	-15	-15
	Mean	-14.99	-15	-14.855	-15
-15.00	Worst	-14.99	-15	-13	-15
	Std. Dev	$0.45e^{-15}$	$3.1\ e^{-7}$	0.459	0
g02	Best	-0.80009	-0.80338	-0.70400	-0.80136
	Mean	-0.79087	-0.79121	-0.56945	-0.75297
-0.803619	Worst	-0.74694	-0.76704	-0.23820	-0.68782
	Std. Dev	0.010912	0.0086	0.0951	0.0320
g03	Best	-1.000	-1.000	-0.461	-1.0
	Mean	-0.999	-1.000	-0.134	-1.0
-1.000	Worst	-0.999	-1.000	-0.002	-1.0
	Std. Dev	$0.37\ e^{-15}$	$3.5\ e^{-4}$	0.117	0
g04	Best	-30665.8217	-30665.539	-30665.539	-30665.5385
	Mean	-30665.8209	-30665.539	-30665.539	-30665.5384
-30665.539	Worst	-30665.8032	-30665.539	-30665.539	-30665.5382
	Std. Dev	0.003391	$1\ e^{-11}$	$e.56\ e^{-6}$	$1\ e^{-4}$
g05	Best	5127.277	5126.498	5126.497	5126.625
	Mean	5127.277	5126.714	5126.497	5378.267
5126.497	Worst	5127.277	5126.824	5126.497	6112.118
	Std. Dev	0	0.43	0	298.01
g06	Best	-6961.813	-6961.814	-6961.814	-6961.813
	Mean	-6961.813	-6961.814	-6961.814	-6961.813
-6961.814	Worst	-6961.813	-6961.814	-6961.814	-6961.813
	Std. Dev	0	$7.1\ e^{-12}$	0	$3.9\ e^{-5}$
g07	Best	24.473	24.307	24.306	24.320
	Mean	26.561	24.317	24.794	24.653
24.306	Worst	29.524	24.356	29.511	25.1347
	Std. Dev	1.642	0.013	1.37	0.219
g08	Best	-0.105	-0.0958	-0.0958	-0.0958
	Mean	-0.105	-0.0958	-0.0958	-0.0958
-0.095825	Worst	-0.105	-0.0958	-0.0958	-0.0958
	Std. Dev	0	$5.8\ e^{-18}$	$4.23\ e^{-17}$	0
g09	Best	680.635	680.630	680.630	680.63
	Mean	680.755	680.634	680.630	680.65
680.630	Worst	680.863	680.646	680.630	680.70
	Std. Dev	0.07932	0.0045	$3.45\ e^{-7}$	0.0167

g10	Best	7049.27	7049.60	7049.248	7050.834
	Mean	7053.21	7077.47	7103.548	8020.7551
7049.248	Worst	7091.88	7183.29	7808.980	9054.2923
	Std. Dev	10.6156	31	148	621.72
g11	Best	0.75	0.75	0.75	0.7499
	Mean	0.75	0.75	0.75	0.7499
0.75	Worst	0.75	0.75	0.75	0.7499
	Std. Dev	0	$3.8\ e^{-6}$	$1.12\ e^{-16}$	0
g12	Best	-1	-1	-1	-1
	Mean	-1	-1	-1	-1
-1	Worst	-1	-1	-1	-1
	Std. Dev	0	0	0	0
g13	Best	0.0555	NA	0.0597	0.0546
	Mean	0.0655	NA	0.3824	0.4588
0.053942	Worst	0.0937	NA	0.9990	0.9949
	Std. Dev	0.0101	NA	0.268	0.3449
g14	Best	-47.453	-47.769	-47.764	-47.517
	Mean	-44.424	-47.750	-47.722	-45.310
-47.7648	Worst	-42.427	-47.712	-47.036	-43.272
	Std. Dev	1.406	0.01	0.162	1.11
g15	Best	961.715	961.715	961.715	961.715
	Mean	961.718	961.715	961.715	963.374
961.715	Worst	961.718	961.716	961.715	970.594
	Std. Dev	$6.87\ e^{-4}$	$3\ e^{-4}$	$2.31\ e^{-13}$	2.275
g16	Best	-1.905154	-1.905155	-1.905155	-1.905155
	Mean	-1.905154	-1.905155	-1.905155	-1.905155
-1.905155	Worst	-1.905154	-1.905155	-1.905149	-1.905155
	Std. Dev	$8.52\ e^{-15}$	$2.4\ e^{-14}$	$1.1\ e^{-6}$	0
g17	Best	8853.539	NA	8853.541	8861.821
	Mean	8853.539	NA	8919.963	8990.997
8853.539	Worst	8853.539	NA	8938.571	9231.201
	Std. Dev	$3.7\ e^{-12}$	NA	25.9	106.19
g18	Best	-0.86463	-0.866025	-0.866025	-0.86596
	Mean	-0.80910	-0.865952	-0.859657	-0.78455
-0.866025	Worst	-0.64443	-0.864843	-0.674981	-0.62977
	Std. Dev	0.06270	$2.1\ e^{-4}$	0.0348	0.09746
g19	Best	32.827	32.725	32.655	33.390
	Mean	35.067	32.952	32.660	38.927
32.655593	Worst	43.374	33.243	32.785	48.487
	Std. Dev	2.2867	0.14	0.0237	3.191

g20	Best	0.21468	NA	NA	NA
	Mean	NA	NA	NA	NA
NA	Worst	NA	NA	NA	NA
	Std. Dev	NA	NA	NA	NA
g21	Best	193.786	NA	193.724	NA
	Mean	193.786	NA	198.090	NA
193.724	Worst	193.787	NA	324.702	NA
	Std. Dev	$3.38\ e^{-5}$	NA	23.9	NA
g22	Best	NA	NA	NA	NA
	Mean	NA	NA	NA	NA
236.430	Worst	NA	NA	NA	NA
	Std. Dev	NA	NA	NA	NA
g23	Best	-400.00	NA	-400.0550	NA
	Mean	-400.00	NA	-392.0296	NA
-400.0551	Worst	-400.00	NA	-342.5245	NA
	Std. Dev	0	NA	12.4	NA
g24	Best	-5.508013	-5.508013	-5.508013	-5.508013
	Mean	-5.508013	-5.508013	-5.508013	-5.508013
-5.508013	Worst	-5.508013	-5.508013	-5.508013	-5.508013
	Std. Dev	$9.46\ e^{-15}$	$1.8\ e^{-15}$	$2.71\ e^{-15}$	0

3.3.2.2. **Les 3 problèmes d'ingénierie**

Les 3 problèmes d'ingénierie considérés sont :

- un problème de conception d'une cuve sous pression (eng 01) [Kannan et Kramer, 1994],
- un problème de conception d'un faisceau soudé (eng 02) [Rao, 1996],
- un problème de tension / compression d'une corde (eng 03) [Belegundu, 1982]

Les problèmes eng01 et eng03 ont été résolus par les approches suivantes : un PSO co-évolutionnaire (CPSO) [He et Wang, 2007a], un PSO hybride (HPSO) [He et Wang, 2007b], une technique d'accélération à compromis adaptatif ATM utilisant une technique de rétrécissement de l'espace de recherche (AATM) [Wang et al, 2009] et une heuristique T-Cell [Aragon et al, 2010].

Le problème g02 a été résolu par les approches suivantes : un algorithme génétique à représentation binaire [Deb, 1991], un modèle de co-évolution basé sur l'algorithme génétique [Coello, 2000], un CPSO [He et Wang, 2007a] et un HPSO [He et Wang, 2007b].

Les résultats statistiques pour les trois problèmes sont présentés dans le Tableau 3-4, le Tableau 3-5 et le Tableau 3-6.

Le nombre de FFE (Fitness Function Evaluations) est utilisé comme critère de comparaison avec les résultats trouvés par d'autres algorithmes. Il correspond au nombre de FFE à partir duquel aucune amélioration notable n'est effectuée : toutes les particules sont pratiquement immobiles. Par exemple, pour le CPSO, la valeur de FFE reportée dans le papier référence a été fixée à 200,000. Pour le CVI-PSO, le nombre de FFE a été fixé à 25,000 car au bout de 25,000 évaluations de la fonction objectif toutes les particules sont pratiquement immobiles et l'algorithme atteint une stabilisation de l'essaim.

Le Tableau 3-4 montre que les meilleures solutions trouvées par le CVI-PSO et le HPSO sont meilleures que celles trouvées par les autres techniques. Toutefois, en comparant les résultats statistiquement, il apparait que les moyennes et ecart-types sont meilleures pour les autres techniques en comparaison au CVI-PSO sauf pour T-Cell qui a les pires performances statistiques. Il est important de noter également que le nombre maximal de FFE est de 81,000 pour le HPSO. Deux contraintes sont actives avec le CVI-PSO alors que le HPSO n'active qu'une seule contrainte. AATM obtient une meilleure solution proche de celle de CVI-PSO avec un écart-type de 4.7 meilleure que celui de CVI-PSO (288.455).

Le Tableau 3-5 montre que la meilleure solution trouvée par le CVI-PSO est comparable à celle obtenue par le HPSO et meilleure que celles obtenues par les autres techniques. La qualité de recherche moyenne du CVI-PSO est supérieure à celles des autres techniques. Même la pire solution trouvée par le CVI-PSO est meilleure que les meilleures solutions trouvées par les autres techniques sauf pour HPSO. L'écart-type des résultats du CVI-PSO est

proche de 0 (6.12 e-4). La meilleure solution trouvée par le CVI-PSO rend actives 4 contraintes.

Du Tableau 3-6 il apparait que la meilleure solution trouvée par le CVI-PSO (0.0126655) est très proche de la meilleure solution du HPSO (0.0126652). Aussi, la qualité de recherche moyenne du CVI-PSO est comparable à celle des autres techniques. En plus, l'écart-type des résultats du CVI-PSO sur 20 exécutions est proche de ceux des autres algorithmes d'optimisation.

Tableau 3-4. Résultats sur le benchmark : Conception d'une cuve sous pression

Algorithme	FFE#	Best	Mean	Worst	Std. Dev
CPSO (He et Wang, 2007a)	200,000	6061.0777	6147.1332	6363.8041	86.45
HPSO (He et Wang, 2007b)	81,000	**6059.7143**	6099.9323	6288.6770	86.20
AATM (Wang et al, 2009)	30,000	6059.7255	**6061.9878**	**6090.8022**	**4.70**
T-Cell (Aragon et al, 2010)	81,000	6390.5540	6737.0651	7694.0668	357.0
CVI-PSO	**25,000**	**6059.7143**	6292.1231	6820.4101	288.45

Tableau 3-5. Résultats sur le benchmark : Conception d'un faisceau soudé

Algorithme	FFE#	Best	Mean	Worst	Std. Dev
GAPF (Deb, 1991)	NA	2.433116	NA	NA	NA
CGA (Coello, 2000)	900,000	1.748309	1.771973	1.785835	0.011
CPSO (He et Wang, 2007a)	200,000	1.728024	1.748831	1.782143	0.012
HPSO (He et Wang, 2007b)	81,000	**1.724852**	1.749040	1.814295	0.040
CVI-PSO	**25,000**	**1.724852**	**1.725124**	**1.727665**	**6.12** e^{-4}

Tableau 3-6. Résultats sur le benchmark : problème de tension / compression d'une corde

Algorithme	FFE#	Best	Mean	Worst	Std. Dev
CPSO (He et Wang, 2007a)	200,000	0.0126747	0.0127300	0.0129240	5.19 e^{-5}
HPSO (He et Wang, 2007b)	81,000	**0.0126652**	**0.0127072**	**0.0127191**	**1.58 e^{-5}**
AATM (Wang et al, 2009)	**25,000**	0.0126682	0.0127080	0.0128613	4.5 e^{-5}
T-Cell (Aragon et al, 2010)	36,000	**0.012665**	0.012732	0.013309	9.4 e^{-5}
CVI-PSO	**25,000**	**0.0126655**	0.0127310	0.0128426	5.58 e^{-5}

3.3.2.3. **Moteur à aimants permanents**

Le modèle du moteur à aimants permanents est testé afin d'évaluer les performances du CVI-PSO. Le modèle est traité sous sa forme reformulée adopté dans le chapitre sur l'optimisation par intervalles.

Plusieurs tests numériques ont été réalisés sur le modèle du moteur sous sa formulation initiale et sous trois autres formulations introduites dans le chapitre 2 sur IBBA. Les caractéristiques de ces modèles sont reportées dans le Tableau 2-3. Les résultats d'optimisation statistiques sont reportés dans le Tableau 3-7 pour les 4 formulations. Dans les tests, on ne considère aucune relaxation des contraintes d'égalité.

La meilleure solution avec une grande précision ($V_u = 6.073466092020043\, e^{-4}$) correspond à l'expérience avec la première reformulation avec un nombre de particules $m = 30$ et un nombre maximal d'itérations $Iter = 200$. Au cours des expérimentations numériques sur le CVI-PSO, il apparait clairement que les deux paramètres m et $Iter$ jouent un rôle crucial dans la qualité de la meilleure solution trouvée. Notons que plus les valeurs de m et $Iter$ augmentent, plus l'optimum global trouvé par le CVI-PSO est affiné en utilisant des calculs de plus haute précision.

En utilisant ces valeurs pour m et $Iter$, l'écart-type des résultats dans 20 exécutions est proche de 0 ($1.36\ e^{-19}$) pour la première reformulation. Ici, le CVI-PSO converge facilement vers l'optimum global avec un nombre pas très élevé d'évaluations. En effet, un optimum moins précis $V_u = 6.073466\, e^{-4}$ peut être atteint avec seulement 5000 évaluations ($m = 30$, $Iter = 50$ et 10 exécutions).

Tableau 3-7. Résultats d'optimisation : modèle initial et reformulations

Stat	Initial	Ref 1	Ref 2	Ref 3
Best (e^{-4})	1.9581	6.0734	6.0734	6.0734
Mean (e^{-4})	9.6559	6.0734	6.1404	6.3089
Worst (e^{-4})	28.860	6.0734	7.0774	10.240
Std. Dev	$7.94\ e^{-4}$	$1.36\ e^{-19}$	$2.54\ e^{-5}$	$7.74\ e^{-5}$
Violation	0.01198	0	0	0

Les résultats obtenus avec la première reformulation sont meilleures que ceux obtenus avec les deux autres. Il est également intéressant de noter que pour toutes les trois reformulations aucune relaxation n'a été nécessaire et la violation des contraintes est nulle avec des contraintes d'égalité actives. Les coefficients α des trois reforulations sont autour de 5%. Malgré le même nombre de degrés de liberté et des valeurs de α proches, le CVI-PSO trouve de meilleures résultats et avec un moindre écart-type avec la première reformulation. Ceci peut s'expliquer par la nature du problème et des contraintes qui n'ont pas les mêmes non-linéarités. Les reformulations sont donc des configurations plus ou moins favorables à la convergence vers l'optimum global et ce, même avec un nombre identique de degrés de liberté.

Dans le cas du problème initial, et avec la même configuration du CVI-PSO, on obtient de meilleures solutions. Ces solutions sont toutefois non réalisables car la violation est toujours non nulle. Ceci s'explique par l'étroitesse des contraintes d'égalités qui les rend difficiles à satisfaire.

Afin d'évaluer l'impact de l'instanciation sur la convergence du CVI-PSO, les deux types d'instanciation développés (Latin-Hypercube et aléatoire) sont testées sur la première reformulation du modèle du moteur à aimants permanents. Les résultats sont reportés dans le Tableau 3-8. Ces deux instanciations sont testées pour différentes tailles de l'essaim et avec 100 exécutions de l'algorithme pour des résultats statistiques plus fiables.

Tableau 3-8. Résultats d'optimisation : échantillonnage aléatoire vs par Latin-Hypercube

Particules #	Aléatoire (e^{-4})				Latin-Hypercube (e^{-4})			
	Best	Mean	Worst	Std.Dev	Best	Mean	Worst	Std.Dev
5	6.0734	8.0815	25.085	3.924	6.0734	7.6032	23.707	3.102
10	6.0734	6.5798	18.765	1.675	6.0734	6.2651	11.787	0.711
20	6.0734	6.1759	6.5376	0.192	6.0734	6.1523	6.5376	0.174
30	6.0734	6.1430	6.5376	0.165	6.0734	6.1198	6.5376	0.139

Le Tableau 3-8 montre l'intérêt de l'utilisation du Latin-Hypercube puisqu'il performe mieux que l'échantillonnage aléatoire quel que soit le nombre de particules. Toutefois, on remarque que plus le nombre de particules augmente et plus l'écart entre les deux types d'échantillonnage se réduit. L'utilisation du Latin-Hypercube est donc plus intéressante pour des essaims plus petits. Ceci va dans le même sens que la réduction du nombre d'évaluations qui est d'autant plus important que les modèles traités sont lourds.

3.4. Contribution : optimisation avec contraintes de type équations différentielles

Dans cette partie, l'algorithme CVI-PSO est étendu pour l'optimisation de modèles dynamiques. Ici, on désigne par modèle dynamique un modèle avec des contraintes de type équations différentielles. Ces contraintes fonctionnelles prennent la forme d'un système d'équations différentielles ordinaires (ODE).

Plusieurs modèles en développement de produits sont dynamiques et intègrent des équations différentielles. En effet, les équations différentielles permettent de prendre en compte la variable temporelle qui traduit un comportement spécifique du produit à concevoir au cours du temps. Ceci permet par exemple de simuler le produit en fonctionnement et de vérifier si des contraintes liées à cette phase sont vérifiées. Ce sont des contraintes issues du cahier des charges, des normes de sécurité etc…

3.4.1. Formulation du problème

Dans cette partie, une formulation mathématique d'un problème d'optimisation dynamique non linéaire est introduite. Dans le contexte présent, le problème d'optimisation est dit dynamique car il contient des contraintes définies par des équations différentielles ordinaires, à ne pas confondre avec la programmation dynamique qui est une méthode de résolution.

Supposons que le problème multi-physique est décrit par le modèle non linéaire algébrique défini précédemment :

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \tag{3.17}$$

Supposons que le comportement temporel du produit est décrit par le système d'équations différentielles ordinaires non linéaires à coefficients non constants suivant :

$$\begin{aligned} \boldsymbol{u}'(t) &= \boldsymbol{h}(\boldsymbol{u}(t), t, \boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{u}(t_0) &= \boldsymbol{u_0}(\boldsymbol{x}, \boldsymbol{y}) \end{aligned} \tag{3.18}$$

Avec :

- t la variable temporelle indépendante, avec $t \in [t_0, t_f] \subset \mathbb{R}$
- t_0 l'instant correspondant à la condition initiale avec $t_0 \in \mathbb{R}$
- $\boldsymbol{u}$: $\boldsymbol{u}(t)$ est le vecteur de dimension p des variables d'état bornées $u_1(t),\ldots, u_p(t)$, avec $\boldsymbol{u}(\tau) \in \left[\underline{u}, \overline{u}\right]^{\tau} \subset \mathbb{IR}^p$ et $\tau \subset [t_0, t_f]$ un ensemble de valeurs de la variables temporelle t.

- $\boldsymbol{h}: \mathbb{R}^p \to \mathbb{R}^p$ est le vecteur de dimension p des équations différentielles ordinaires non linéaires $h_1,\ldots, h_p$. $\boldsymbol{h}$ est un vecteur champ de classe $\mathcal{C}^1$.

$\boldsymbol{h}$ peut aussi dépendre de y, mais sachant que $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x})$ et pour des raisons de simplification on considère que $\boldsymbol{h}$ dépend des paramètres d'entrée $\boldsymbol{x}$. Le problème dynamique paramétré est donc formulé comme suit :

$$\begin{aligned} &\boldsymbol{u}'(t) = \boldsymbol{h}(\boldsymbol{u}(t), t, \boldsymbol{x}) \\ &\boldsymbol{u}(t_0) = \boldsymbol{u_0}(\boldsymbol{x}) \end{aligned} \tag{3.19}$$

Dans ce travail, le modèle dynamique et donné sous forme autonome :

$$\begin{aligned} &\boldsymbol{u}'(t) = \boldsymbol{h}(\boldsymbol{u}(t), \boldsymbol{x}) \\ &\boldsymbol{u}(t_0) = \boldsymbol{u_0}(\boldsymbol{x}) \end{aligned} \tag{3.20}$$

Un système non autonome peut facilement être reformulé en système autonome en considérant la variable temporelle comme variable d'état de dérivée égale à 1. Ainsi, le système non autonome $u'(t) = u(t) + 2t$ peut être réécrit comme $u_1'(t) = u_1(t) + 2u_2(t)$ avec $u_2'(t) = 1$.

Le problème d'optimisation est donc formulé de la manière suivante :

$$\begin{cases} \min\limits_{x \in X \subseteq \mathbb{R}^n} \phi(\boldsymbol{u}(t), \boldsymbol{x}) \\ \boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \\ \boldsymbol{u}'(t) = \boldsymbol{h}(\boldsymbol{u}(t), \boldsymbol{x}) \\ \boldsymbol{u}(t_0) = \boldsymbol{u_0}(\boldsymbol{x}) \\ \boldsymbol{u}(\tau) \in \left[\underline{u}, \overline{u}\right]^{\tau}, \boldsymbol{x} \in \left[\underline{x}, \overline{x}\right], \boldsymbol{y} \in \left[\underline{y}, \overline{y}\right] \\ t \in \left[t_0, t_f\right], \qquad \tau \subset [t_0, t_f] \end{cases} \tag{3.21}$$

Avec :

- ϕ la fonction objectif qui peut dépendre de $\boldsymbol{u}(t)$ et de $\boldsymbol{x}$
- $\boldsymbol{u}(\tau) \in \left[\underline{u}, \overline{u}\right]^{\tau}$ l'ensemble des contraintes du type équation différentielle à respecter

Un exemple de modèle dynamique d'optimisation d'un actionneur différentiel a été présenté dans le paragraphe 1.2.2. Ce modèle servira de base pour les tests numériques de l'algorithme qui sera introduit.

3.4.2. Algorithme proposé : PSO-RK44

L'algorithme proposé est basé sur l'algorithme CVI-PSO développé dans la partie précédente. Il est adapté afin de prendre en compte les contraintes du type ODE.

Etant donné que PSO peut traiter le modèle comme une boite noire, les méthodes de résolution numériques d'équations différentielles sont les plus adaptées. En effet, nous proposons d'intégrer l'algorithme Runge-Kutta 44 de résolution numérique d'ODE dans PSO. PSO étant présenté dans la partie précédente, nous présenteront l'algorithme Runge-Kutta 44 (RK44) et le schéma global de l'algorithme d'optimisation.

Runge-Kutta 44 est un algorithme largement diffusé en ingénierie et qui dispose d'un bon compromis entre la précision et la rapidité [Bird J. O., 2010][3]. Soit l'équation différentielle suivante :

$$\begin{aligned} &\frac{dx}{dt} = f(x,t) \\ &x(0) = x_0 \end{aligned} \tag{3.22}$$

Avec $f: \mathbb{R}^{d+1} \to \mathbb{R}^d$ un vecteur champ de classe $\mathcal{C}^1$ and $t \in [0, t_f]$.

L'objectif est de trouver la fonction x à l'instant t_f. Le principe de fonctionnement de Runge-Kutta 44 consiste à discrétiser le temps avec un pas h et estimer la valeur de la fonction aux instants nh. La formule de Runge-Kutta 44 est la suivante :

$$x((n+1)h) = x(nh) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.23}$$

Avec :

$k_1 = hf(nh, x(nh))$

$k_2 = hf((n+\frac{1}{2})h, x(nh) + \frac{1}{2}k_1)$

$k_3 = hf((n+\frac{1}{2})h, x(nh) + \frac{1}{2}k_2)$

$k_4 = hf((n+1)h, x(nh) + k_3)$

Runge-Kutta 44 demande 4 évaluations de f à chaque pas de temps. C'est une méthode à un pas qui ne nécessite pas les dérivées de f et facile à implémenter.

L'algorithme PSO-RK44 est composé de deux parties : une partie qui permet le calcul de la fonction objectif et les valeurs des contraintes, et une partie qui gère la partie optimisation. Le

[3] Higher engineering mathematics

PSO joue le deuxième rôle en utilisant le mécanisme de gestion de contraintes du CVI-PSO présenté dans la partie précédente.

L'algorithme fonctionne en deux parties séparées. La première partie représente le modèle avec ses deux composantes dynamique et statique. La deuxième partie est l'algorithme d'optimisation qui va piloter le calcul.

La première partie est illustrée par la Figure 3-3 où on voit qu'à partir d'une solution de conception X, les performances sont calculées. Ceci nécessite la résolution de la partie statique et de la partie dynamique.

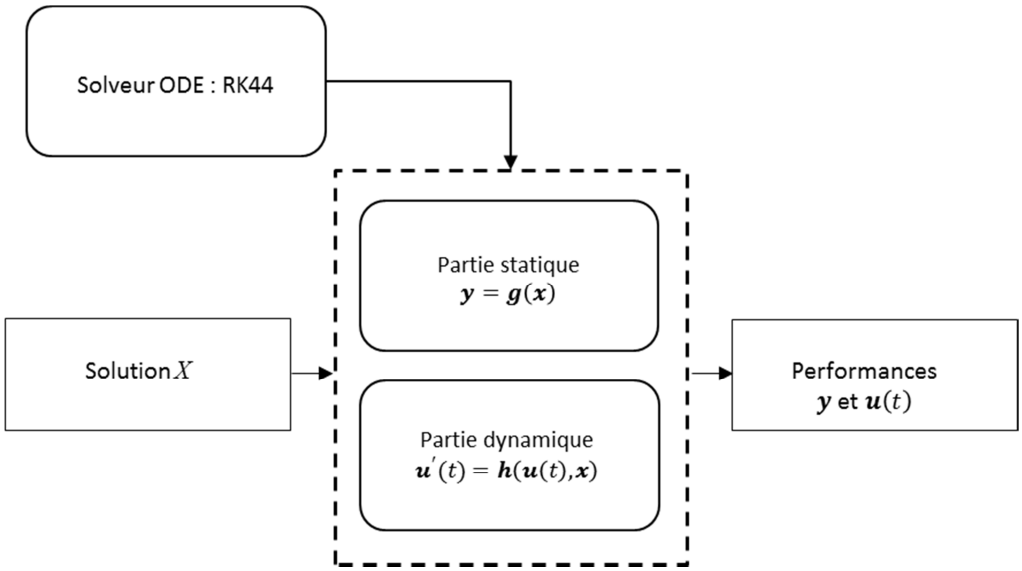

Figure 3-3. PSO-RK44 : gestion du modèle dynamique

La deuxième partie (voir Figure 3-4) concerne l'optimisation. Cette étape consiste à vérifier la faisabilité via le mécanisme de gestion de contraintes et générer les nouvelles solutions (nouvelles particules dans CVI-PSO).

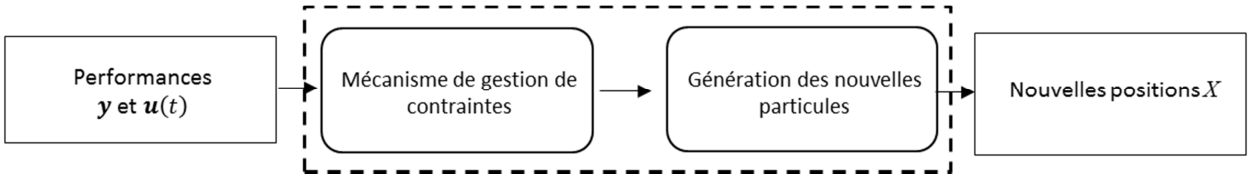

Figure 3-4. PSO-RK44 : étapes d'une itération

En réalité, ces deux parties sont liées et sont appelées à chaque itération de l'algorithme d'optimisation. Le schéma global représenté par la Figure 3-5 montre cette interaction. En effet, à chaque itération du CVI-PSO et pour chacune des particules, les contraintes analytiques et les paramètres de l'ODE sont calculés. Ensuite l'algorithme Runge-Kutta 44

permet de résoudre l'ODE et ainsi déterminer les paramètres contraints de l'ODE. Par la suite, les sorties de l'ODE sont injectées dans le mécanisme de gestion de contraintes de CVI-PSO afin de mettre à jour les meilleures positions de tout l'essaim xg et de chaque particule xp^j et mettre à jour les positions des particules.

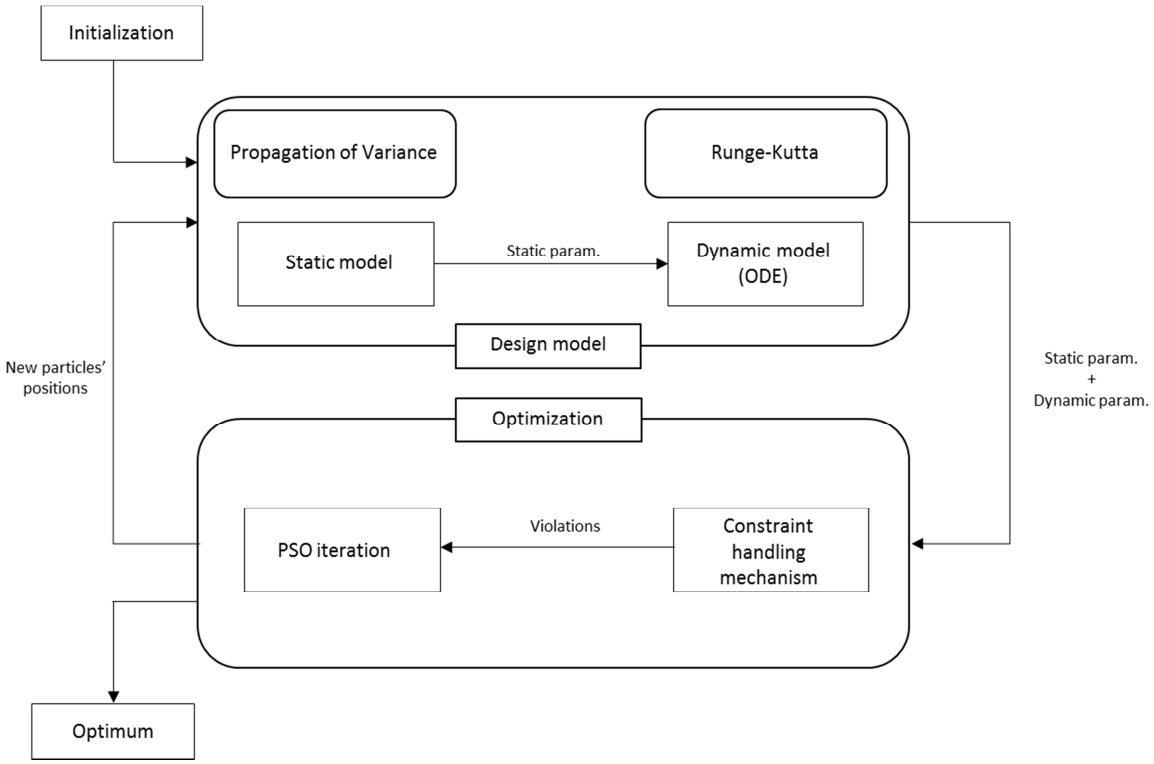

Figure 3-5. PSO-RK44 : algorithme global

Algorithme 3-2. PSO avec le mécanisme de gestion de contraintes proposé

1: Choose a swarm size m, a stopping criterion and PSO parameters (c_1& c_2)

2: Initialization ($t = 0$)

For j from 1 to m

3: randomly generate an initial particle position $x^j(0)$ and velocity $v^j(0)$

4: compute the constraints values $g_k(\ x^j(0))$

5: Solve $\boldsymbol{u}'(t) = \boldsymbol{h}\left(\boldsymbol{u}(t), x^j(0)\right)$ and compute the constraints values $\boldsymbol{u}(\tau)$

6: compute the total constraints violation $dp(\ x^j(0))$

7: update the particle's best position : $xp^j(0) := x^j(0)$

End For

8: update the initial global best position $xg(0)$

Repeat

9: $t := t + 1$

For j from 1 to m

10: update particle's velocity $v^j(t)$ (Equ. 3.2) and position $x^j(t)$ (Equ. 3.3)

11: enforce the bound constraints (Equ. 3.6)

12: compute the constraints values $g_k(\ x^j(t))$

13: Solve $\boldsymbol{u}'(t) = \boldsymbol{h}\left(\boldsymbol{u}(t), x^j(t)\right)$ and compute the constraints values $\boldsymbol{u}(\tau)$

14: compute the total constraints violation $dp(\ x^j(t))$

15: update the particle's best position $xp^j(t)$

End For

16: update the global best position $xg(t)$

Until stopping criterion is met

3.4.3. Tests numériques : actionneur différentiel

L'algorithme CVI-PSO-RK44 a été implémenté et testé en Java et Matlab version 7.1. Les tests ont utilisé des populations de différentes tailles m = 30, 50 et 100 particules car ceci a semblé suffisant pour le problème de conception du déclencheur électrique.

CVI-PSO-RK44 est exécuté et le nombre d'itérations nécessaire pour atteindre la stabilité de l'essaim est reporté dans Iter#. 20 essais sont réalisés pour chaque configuration et les performances statistiques sont calculées sur leur base.

Les résultats statistiques de simulation sont reportés dans le Tableau 3-9. Ceux-ci incluent les meilleures/pires valeurs trouvées, leurs moyennes et écart-types. Le nombre total d'évaluation du modèle FFE est défini par $FFE = m \times iter\# \times 20$.

Tableau 3-9. Résultats d'optimisation : Actionneur différentiel

Particules #	Iter. #	FFE	Performances (e^{-6})			
			Best	Worst	Mean	Std. Dev
30	140	84000	1.511	32.83	6.537	7.82
50	48	48000	1.508	21.00	3.664	4.41
100	50	100000	1.468	6.924	2.388	1.42

La Figure 3-6 illustre le processus d'évolution typique de la fonction objectif en résolvant le problème avec 30, 50 et 100 particules. C'est un problème de minimisation sous contraintes, ce qui explique l'augmentation de la valeur de la solution avec les itérations. En effet, les premières solutions sont très performantes mais non faisables du point de vue des contraintes. Plus les itérations sont exécutées, et plus les solutions s'approchent de la zone faisable du domaine de recherche au détriment de la fonction objectif.

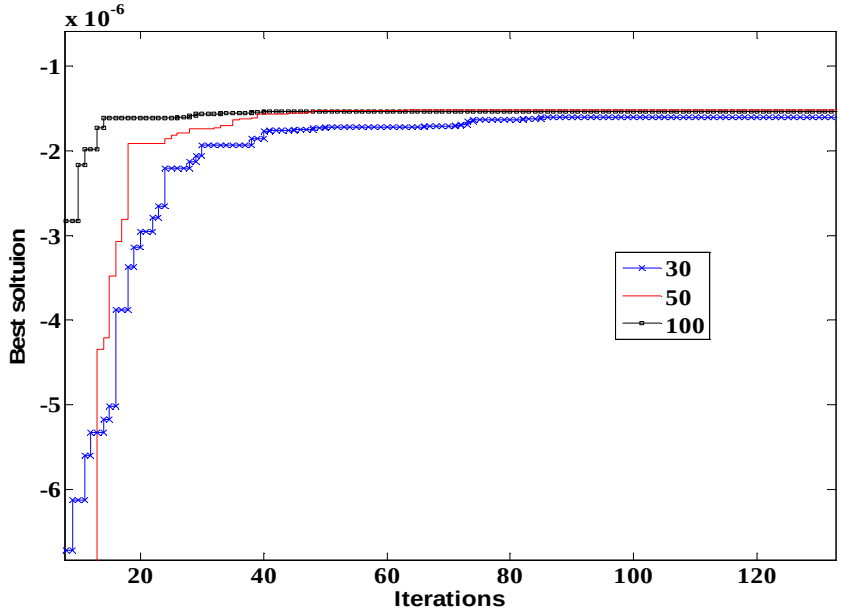

Figure 3-6. Evolution de la fonction fitness avec 30, 50 et 100 particules

Dans tous les tests réalisés, des solutions faisables ont été trouvées et les meilleures solutions sont proches. Ce sont des solutions qui respectent aussi bien les contraintes algébriques que les contraintes de type équations différentielles. Comme le montre le Tableau 3-9, la meilleure solution, la pire, la moyenne et l'écart-type sont améliorés en augmentant la taille de l'essaim. L'exploration de l'espace de recherche est effectuée par les particules qui assurent une recherche d'autant plus globale que leur nombre est grand. Ainsi, l'exploration est d'autant plus meilleure que la taille de l'essaim est plus grande. L'exploration permet d'améliorer sensiblement les solutions et ainsi atteindre l'optimum global rapidement grâce à un bon réglage des paramètres du CVI-PSO. Ceci explique la réduction de FFE# lorsque la taille de l'essaim augmente de 30 à 50. Une fois la limite de l'exploration atteinte, les particules de l'essaim concentrent leur recherche dans des zones très localisées : c'est l'intensification. C'est ce qui explique l'augmentation du nombre de FFE tout en améliorant légèrement la solution en passant de 50 à 100 particules.

Les trois meilleures solutions obtenues ont été utilisées en tant que points de départ de l'algorithme d'optimisation local SQP (Sequential Quadratic Programming). Nous avons constaté que toutes ces trois solutions permettent de converger vers le même optimum qui est $1.467.10^{-6}$ ce qui est très proche des solutions du CVI-PSO-RK44. Sachant que PSO-RK44 ne demande aucun point de départ, les solutions obtenues sont très intéressantes.

3.5. Conclusion

Ce chapitre a fait l'objet d'une méthode d'optimisation stochastique permettant de couvrir un large éventail de problèmes de dimensionnement. Cette méthode est basée sur l'algorithme d'optimisation par essaims particulaires ou PSO.

Dans cette partie, deux contributions ont été présentées. La première consiste en un nouveau mécanisme de gestion de contraintes pour l'algorithme PSO sous contraintes. Ce mécanisme est utilisé lors de la mise à jour de la meilleure position d'une particule et celle de tout l'essaim. L'idée est de transformer le problème d'optimisation sous contraintes en un problème bi-objectif non contraint résolu par une méthode lexicographique. A chaque itération du CVI-PSO, une violation totale des contraintes est mesurée et normalisée grâce à l'arithmétique d'intervalles. Cette violation est utilisée comme deuxième objectif à minimiser. Sachant que la violation est formulée de telle sorte à ce qu'elle soit toujours positive, le CVI-PSO privilégie l'annulation de la valeur de la violation même en ''détériorant'' la fonction objectif. Une fois les particules dans la région faisable (violation = 0), le CVI-PSO privilégie la minimisation de la fonction objectif.

Plusieurs tests numériques ont été réalisés afin d'évaluer les performances de l'algorithme et de le comparer à d'autres algorithmes existants. Ces tests ont été réalisés sur un benchmark de 24 problèmes d'optimisation mathématiques et 3 problèmes d'ingénierie. Ces tests ont démontré l'intérêt de notre algorithme surtout pour résoudre des problèmes d'ingénierie. Pour plusieurs problèmes du benchmark, les solutions du CVI-PSO sont comparables aux meilleures solutions connues tout en activant les contraintes d'égalité et avec un écart-type proche de zéro. Par ailleurs, un autre critère doit être mis en évidence qui est la simplicité d'implémentation du CVI-PSO et le nombre réduit de paramètres de réglage.

La deuxième contribution est une extension de l'algorithme CVI-PSO pour traiter des problèmes de dimensionnement avec des contraintes du type équations différentielles. Ceci est réalisé en intégrant une méthode de résolution numérique d'équations différentielles (Runge Kutta) dans la boucle d'optimisation. Une fois les valeurs des contraintes calculées, le mécanisme de gestion de contraintes introduit précédemment permet de classer les particules par ordre de préférence et de générer les nouvelles positions. Les tests numériques sur l'actionneur différentiel ont montré l'efficacité de cet algorithme.

Les contributions de ce chapitre ont fait l'objet de deux papiers dans le journal Engineering Application of Artificial Intelligence [Mazhoud I. et al, 2013] et d'une participation à la conférence internationale IEEE CCECE [Mazhoud I. et al, 2012d].

Les algorithmes présentés jusque-là permettent de faire de l'optimisation classique. Les modèles traités sont les modèles algébriques et les modèles avec des contraintes de type ODE. Le prochain chapitre est consacré à l'optimisation robuste. En effet, une méthode d'optimisation permettant de prendre en compte les variabilités sur les paramètres de conception sera introduite dans le prochain chapitre. Elle permettra de traiter les modèles algébriques et les modèles avec des contraintes de type ODE.

4. Optimisation robuste : des modèles algébriques aux modèles dynamiques

En conception de produits, maîtriser les incertitudes est un enjeu très important et peut conduire à des gains substantiels en coûts et en temps. Ne pas prendre en compte les incertitudes peut avoir plusieurs conséquences comme les problèmes d'assemblage, de non-conformités et de performances non contrôlées. Ceci conduit à avoir des rebus en sortie de production. L'optimisation robuste vise à prendre en considération les variabilités au cours du processus d'optimisation et donc minimiser les rebus lors de la production en série. Il est donc important de prendre en compte ces variabilités le plus en amont possible afin de réduire au maximum leurs effets. L'objectif de ce chapitre est de traiter des problèmes d'optimisation en présence d'incertitudes sur les variables. L'idée est de partir d'un modèle sans incertitudes et de générer automatiquement un modèle reformulé qui prend en compte les incertitudes (Figure 4-1). Ce modèle reformulé permet de calculer les variabilités en sortie de modèle en fonction des variabilités en entrée. La reformulation porte également sur le cahier des charges. Elle vise à reformuler la fonction objectif et les contraintes de telle sorte à orienter l'algorithme d'optimisation vers les solutions robustes. Nous travaillerons avec l'algorithme CVI-PSO introduit précédemment pour l'optimisation.

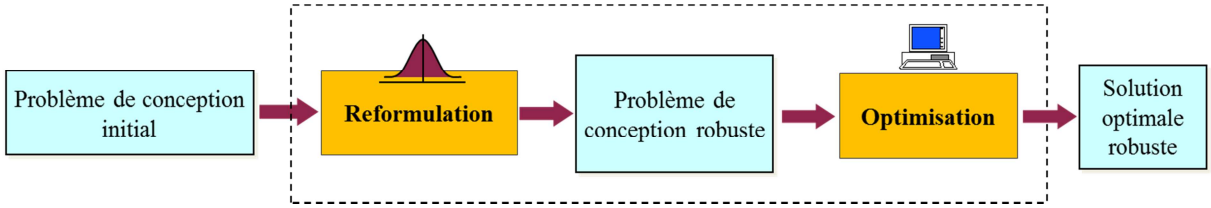

Figure 4-1. Démarche de l'optimisation robuste

Dans un premier lieu, le choix de modélisation des incertitudes et du cahier des charges robuste sera introduit. Puis deux méthodes de propagation d'incertitudes seront introduites et comparées. Ensuite, une extension de la méthode de propagation d'incertitudes Propagation of Variance (PoV) aux modèles dynamiques contenant des équations différentielles sera présentée. Enfin, des tests sur l'optimisation robuste de l'actionneur différentiel seront présentés.

4.1. Incertitudes et robustesse

Dans les chapitres précédents, nous avons travaillé avec deux hypothèses implicites. La première suppose que les modèles sont exactes et représentent fidèlement le comportement du produit. La deuxième hypothèse suppose que les paramètres de conception ne sont pas assujettis à des variabilités et peuvent donc être considérés comme mono-valuées. Ces deux hypothèses sont en réalité fausses et conduisent à introduire deux types d'incertitudes :

- Incertitudes sur le modèle : le modèle utilisé qui simule le comportement réel du produit n'est jamais exact. Ceci peut être dû au manque de connaissance des phénomènes mis en œuvre mais aussi à une volonté de simplification et d'abstraction. En effet prendre en compte tous les paramètres conduit à un modèle lourd en calcul. Le modélisateur est amené à faire un compromis entre la précision et la rapidité ce qui conduit nécessairement à des simplifications.
- Incertitudes sur les paramètres : les paramètres de conception sont soumis à des variabilités. Ceci est dû au manque de connaissance sur les paramètres environnementaux (température, coût des matières dans le marché mondial …). Le processus de fabrication conduit à des dérives et des variabilités au niveau de la géométrie, assemblage …etc. De plus, les propriétés du produit peuvent varier au cours du temps (vieillissement des matériaux plastiques par exemple).

Dans nos travaux, on fait l'hypothèse que le modèle est exact. On considère donc que le modèle de conception a un bon niveau de précision et que ses variabilités sont négligeables devant les variabilités sur les paramètres de conception.

Dans cette étude, la conception robuste est utilisée comme suit :

- « Si la conception garantit les performances du cahier des charges fonctionnel quelles que soient les variabilités des paramètres alors la conception est dite robuste » [Klein Meyer et al, 2007]
- « Concevoir un produit moins sensible aux incertitudes plutôt que de supprimer les causes de ces variations » [Huang and Du, 2006]

Deux exigences se dégagent de ces définitions. La première est la garantie que les contraintes du cahier des charges sont respectées quelles que soient les variabilités (aux défauts près). La deuxième est d'avoir des performances moins sensibles aux variabilités.

On considère le modèle d'optimisation suivant :

$$\begin{cases} \min\limits_{x \in X \subseteq \mathbb{R}^n} f(x) \\ y = g(x) \\ x \in [\underline{x}, \overline{x}], y \in [\underline{y}, \overline{y}] \end{cases}$$

Plusieurs approches existent pour modéliser les variabilités des paramètres de conception. On peut citer la modélisation par intervalles [Moore R.E, 1966], par ensembles flous [Zadeh L. A.,1965], ou par variables aléatoires [Rolander et al., 2006].

La modélisation par intervalles suppose que chaque paramètre est caractérisé par une limite inférieure et une limite supérieure. Ainsi, le paramètre de conception x évolue entre une limite inférieure x_{inf} et une limite supérieure x_{sup}:

$$x \leftarrow [x_{inf}, x_{sup}] \qquad (4.1)$$

Cette modélisation a l'avantage de la simplicité mais néglige la répartition du paramètre x entre les limites x_{inf} et x_{sup}. En présence de modèle algébrique, le calcul des limites des paramètres de sortie peut être assuré par l'arithmétique d'intervalle introduite précédemment [Moore R.E, 1966]. Cette arithmétique permet de calculer de façon garantie les intervalles des sorties en fonction des intervalles des entrées via les fonctions d'inclusion. Toutefois, cette approche ne fournit aucune information sur la probabilité d'occurrence d'un scénario en particulier.

Un ensemble classique repose sur des fonctions d'appartenance binaires. En effet, pour un sous ensemble A d'un ensemble X, la fonction d'appartenance au sous ensemble A d'un élément x consiste à retourner la valeur 1 si $x \in A$ et 0 si $x \notin A$. Les ensembles flous reposent sur des fonctions d'appartenance non binaires. La fonction d'appartenance au sous ensemble flou A d'un élément x attribue une valeur entre 0 et 1 en fonction de la certitude du scénario $(x \in A)$ [Zadeh L. A., 1965]. De la même façon que les intervalles, une arithmétique floue a été introduite qui est par ailleurs complexe. En fait, les ensembles flous disposent des opérateurs égalité, union, inclusion, intersection mais aussi de l'addition, la multiplication,… Cette arithmétique nécessite une « fuzzification » pré-calcul et une « dé-fuzzification » post-calcul ce qui fait perdre de l'information et rend le calcul complexe [Liu X., 2007].

La modélisation probabiliste est certainement la plus répandu en conception robuste [Rolander et al., 2006]. Elle consiste à modéliser les incertitudes sur les paramètres de conception par des distributions aléatoires. La fonction de densité de probabilité permet de calculer la probabilité que la variable x soit comprise dans l'intervalle $[a, b]$:

$$P(a \leq x \leq b) = \int_a^b f(x)dx \tag{4.2}$$

Plusieurs types de densité de probabilité existent : normale, uniforme, exponentielle… Cette modélisation requiert d'une part un nombre de calcul souvent plus important, d'autre part nécessite plus d'informations sur les paramètres qu'une modélisation par intervalles. Il faut en effet connaître l'allure de la dispersion du paramètre autour de sa valeur nominale. La difficulté de cette modélisation est de disposer du type de distribution pour chaque paramètre. Cet inconvénient se révèle plus important encore dans les premières phases de conception où la quantité d'information est plus faible. Des distributions normales sont souvent utilisées pour pallier à ce manque d'information [Picheral L., 2013].

On a fait ici le choix de modéliser les paramètres de conception par leurs distributions de probabilité indépendantes caractérisées par une moyenne μ et un écart-type σ. Le choix s'est arrêté aux deux premier moments d'une part parce que ceci permet d'avoir des calculs plus rapides et d'autre part parce qu'avec les outils disponibles, il n'est pas possible de faire le calcul exacte des dérivées d'ordre supérieur. En effet, ce choix permet d'avoir plus d'informations sur les paramètres de performances du produit sans la perte d'informations. De plus, plusieurs méthodes existent pour la propagation d'incertitudes avec des paramètres probabilistes. La modélisation probabiliste demande un calcul plus coûteux et nécessite

parfois plus de données sur les paramètres de conception que les autres modélisations. Cependant, elle fournit des résultats porteurs de plus d'informations qu'un intervalle ou qu'un ensemble flou. Les résultats sont par ailleurs exploitables directement contrairement aux ensembles flous qui nécessitent une « dé-fuzzification » [Picheral L., 2013].

L'optimisation robuste nécessite de reformuler le modèle de comportement multi-physique et le cahier des charge. La reformulation du modèle multi-physique rend le calcul des variabilités des sorties en fonction des variabilités des entrées possible. La reformulation du cahier des charges permet d'orienter l'algorithme d'optimisation vers les solutions les plus robustes (Figure 4-2).

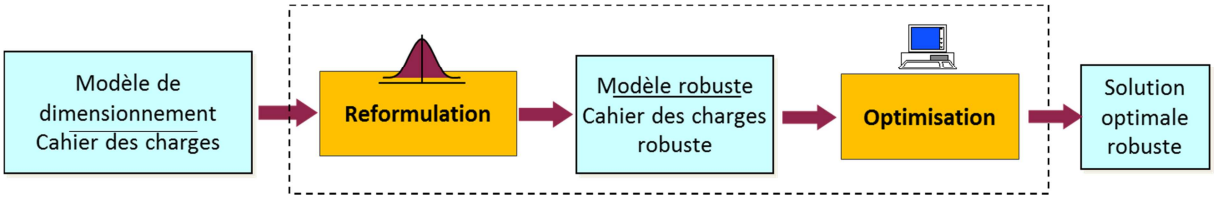

Figure 4-2. Optimisation robuste : reformulation

Afin de prendre en compte la robustesse dans l'optimisation, plusieurs formulations de la fonction objectif et des contraintes ont été introduites.

[Lee. K., Park. G., 2001] proposent une formulation de la fonction objectif consistant en une pondération entre la moyenne μ_f et l'écart-type σ_f :

$$f_r(\mu_x, \sigma_x) = \alpha . \frac{\mu_f}{{\mu_f}^*} + (1 - \alpha) . \frac{\sigma_f}{{\sigma_f}^*} \tag{4.3}$$

Les contraintes en sortie $y \in \left[\underline{y}, \overline{y}\right]$ sont transformées de la façon suivante :

$$[y - k \sum_{i=1}^{n} \left|\frac{\partial g}{\partial X_i}\right| \Delta X_i, y + k \sum_{i=1}^{n} \left|\frac{\partial g}{\partial X_i}\right| \Delta X_i] \subseteq \left[\underline{y}, \overline{y}\right] \tag{4.4}$$

avec :

- f_r est la fonction objectif robuste reformulée.
- μ_f et σ_f sont respectivement la moyenne et l'écart-type de la fonction objectif f du cahier des charges initial.
- ${\mu_f}^*$ et ${\sigma_f}^*$ sont respectivement les valeurs optimales de la moyenne et de l'écart-type de la fonction objectif initiale f optimisés séparément.
- α est un poids fixé entre 0 et 1. Une valeur proche de 0 signifie que l'on privilégie la minimisation de l'écart-type par rapport à la moyenne, et inversement pour une valeur proche de 1.

- k est le facteur de pénalité avec le respect de chaque contrainte déterminé par le concepteur. C'est en réalité le niveau de variation admissible pour la contrainte.

Cette formulation transforme la fonction objectif en fonction objectif robuste. La fonction objectif robuste minimise la moyenne et l'écart-type de la fonction objectif avec un poids attribué à chacun d'entre eux. La formulation des contraintes a pour objectif de maîtriser la faisabilité avec un facteur de pénalité k.

[Du X., et al, 2009] suggèrent une formulation basée sur l'approche 6 sigmas pour les contraintes et une pondération entre la moyenne et l'écart-type pour la fonction objectif :

$$\begin{cases} \underset{x \in X \subseteq \mathbb{R}^n}{min}\, w_1 \mu_f + w_2 \sigma_f \\ y = g(x) \\ x \in [\underline{x}, \overline{x}], [\mu_y - k\sigma_y; \mu_y + k\sigma_y] \subseteq [\underline{y}, \overline{y}] \end{cases} \quad (4.5)$$

avec :

- μ_f et σ_f : respectivement la moyenne et l'écart-type de la fonction objectif f
- w_1 et w_2 : les poids donnés respectivement à la moyenne μ_f et à l'écart-type σ_f
- μ_y et σ_y : respectivement la moyenne et l'écart-type de y
- k est le niveau de variation toléré pour la contrainte initiale g. Si $k = 3$ par exemple, cela signifie que l'on souhaite que 99.73% des contraintes initiales $y \in [\underline{y}, \overline{y}]$ soient satisfaites.

Dans cette formulation, [Du X., et al, 2009] suggèrent d'utiliser une fonction objectif pondérée entre la moyenne et l'écart-type. Toutefois, et contrairement à l'approche précédente [Lee. K., Park. G., 2001], ils ne proposent pas de normalisation ce qui peut rendre le choix des coefficients w_1 et w_2 difficile. En effet, la moyenne et l'écart-type ont souvent des ordres de grandeur très différents. Les contraintes sont formulées avec une approche qui permet de garantir la faisabilité de la solution optimale avec un pourcentage contrôlé (6 sigmas).

Pour plus de détails à propos des différentes formulations voir [Picheral L., 2013] [Picheral L., Mazhoud I. et al, 2013]. La reformulation adoptée est une combinaison des différentes formulations introduites. On considère cette fonction objectif :

$$f_r(\mu_x, \sigma_x) = \alpha . \frac{\mu_f}{{\mu_f}^*} + (1 - \alpha) . \frac{\sigma_f}{{\sigma_f}^*} \quad (4.6)$$

avec μ_f et σ_f respectivement la moyenne et l'écart-type de f. ${\mu_f}^*$ et ${\sigma_f}^*$ sont respectivement les valeurs optimales de μ_f et σ_f obtenues en optimisant un des deux termes. Le paramètre α est un paramètre de pondération qui permet de travailler en mono-objectif avec $0 \leq \alpha \leq 1$.

Afin de prendre en compte les variabilités sur les paramètres de conception et les paramètres de performances, $x \in [\underline{x}, \overline{x}]$ et $y \in [\underline{y}, \overline{y}]$ sont transformés en :

$$\begin{aligned}[\mu_x - k\sigma_x; \mu_x + k\sigma_x] &\subseteq [\underline{x}, \overline{x}] \\ [\mu_y - k\sigma_y; \mu_y + k\sigma_y] &\subseteq [\underline{y}, \overline{y}]\end{aligned} \tag{4.7}$$

où μ_x, μ_y, σ_x et σ_y les moyennes et écart-types des paramètres d'entrée et de sortie. Ce choix permet de maîtriser le taux de non-conformité en choisissant la valeur de k. Dans les tests numériques, nous utilisons l'approche 6σ en prenant $k = 3$.

Si on suppose que le modèle représente fidèlement la réalité et que les distributions des paramètres d'entrées sont justes, l'approche 6σ signifie que l'on garantit que 99,73% des produits fabriqués respecteront chaque contrainte. Les contraintes étant indépendantes, il y aura 0.27% de rebus pour chaque contraintes. S'il y'a n contraintes, cela signifit qu'au maximum il y'aura $n.0.27\%$ de rebuts total.

L'objectif de la reformulation est d'intégrer les variabilités dans l'objectif à optimiser et les contraintes à respecter. Les méthodes de propagation d'incertitudes permettent de calculer les variabilités en sortie en fonction des variabilités en entrée.

La propagation d'incertitudes consiste à remplacer le modèle en valeur nominales g par un modèle qui prend en entrées les Moments des paramètres d'entrée et qui donne en sortie les Moments des paramètres de sortie. Ce modèle sera appelé g_r :

$$(\mu_Y, \sigma_Y) = g_r(\mu_X, \sigma_X) \tag{4.8}$$

Le prochain paragraphe introduit le choix de la méthode de propagation d'incertitudes adoptée.

4.2. Propagation des incertitudes

La propagation d'incertitudes a pour objectif de donner une mesure des variabilités des paramètres de sortie en fonction des variabilités des paramètres d'entrée. On considère que les variabilités sont mesurées par la moyenne et l'écart-type. La Figure 4-3 résume les étapes de l'optimisation robuste.

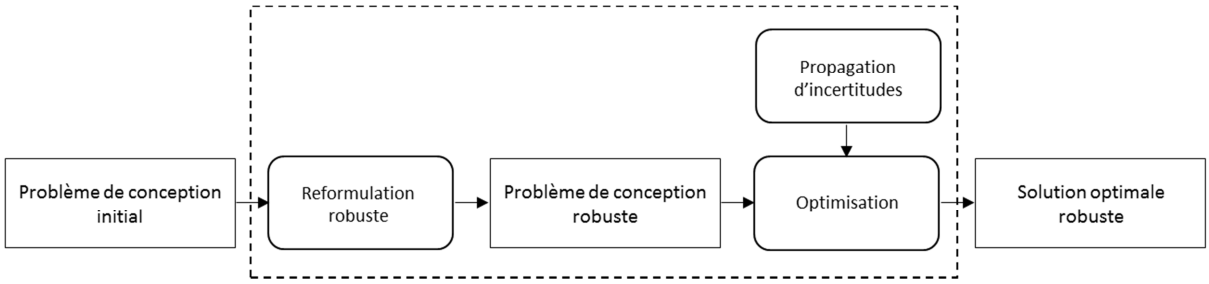

Figure 4-3. Etapes de l'optimisation robuste

Considérons $x = [x_1, \dots, x_n]$ le vecteur des variables aléatoires d'entrée $(x_i)_{1 \le i \le n}$. Soit f_{x_i} la fonction densité de probabilité de la variable x_i. Ainsi, l'espérance (moyenne) de x_i est donnée par :

$$\mu_{x_i} = E[x_i] = \int_{\mathbb{R}} x f_{x_i}(x) dx \tag{4.9}$$

L'écart-type de x_i est donné par la formule suivante :

$$\sigma_{x_i}^2 = E[(x_i - E[x_i])^2] = E[x_i^2] - (E[x_i])^2 \tag{4.10}$$

Les quantités μ_{x_i} et σ_{x_i} sont supposées connues pour tout $1 \le i \le n$. C'est là que les méthodes de propagation d'incertitudes interviennent pour d'évaluer μ_{y_i} et σ_{y_i} pour $1 \le i \le m$.

Plusieurs méthodes existent dans la littérature avec différentes propriétés [Amelin, 2004] [Forrester and Keane, 2009] [Glancy, 1999]. Deux méthodes très utilisées et une comparaison de leurs propriétés seront présentées : la méthode Monte-Carlo [Amelin, 2004] et la méthode Propagation of Variance (PoV) [Glancy, 1999]. Des travaux antérieurs ont mis en évidence les propriétés de plusieurs méthodes de propagation d'incertitudes et ont montré l'intérêt d'utiliser la méthode PoV en optimisation robuste (Tableau 4-1).

Tableau 4-1. Méthodes de propagation d'incertitudes : caractéristiques [Picheral. L., 2013]

		Modélisation des paramètres d'entrée	**Pré-traitement des données en entrée**	**Type de résultat pour les paramètres de sortie**	**Calcul des dérivées**	**Type de lien entre variation des entrée et des sorties**	**Nombre d'appel au modèle**	**Type d'optimisation**
Simulations de Monte-Carlo	10^6 sur modèle initial	Distribution complète	Aucun	Distribution complète	Non	Numérique	10^6	Stochastique
	10^6 sur modèle approché	Distribution complète	Aucun	Distribution complète	Oui	Numérique	10^6 + nombre d'appel au modèle pour obtenir le méta-modèle	Stochastique
	100 à 1000 Latin Hypercube	Distribution complète	Aucun	Distribution complète	Non	Numérique	100 à 1000	Stochastique
Propagation of Variance	1er ordre	Moments (moyenne et écart-type)	Oui pour obtenir une loi normale approchée	Moments (Moyenne et écart-type)	Gradient	Numérique si calcul dérivées numérique, analytique si calcul formel	$2n + 1$ (ou n est le nombre de paramètre d'entrée)	Stochastique si calcul dérivées numérique ou déterministe si calcul formel
	2nd ordre				Gradient et Hessian		$2n^3 + 3n$ (par différence finie)	
Univariate Dimension Reduction		Distribution complète	Aucun	Moments ordinaires d'ordre n	Non	Numérique	$3n + 1$ avec quadrature	Stochastique
Polynômes de Chaos		Distribution complète	Aucun	Moments ordinaires d'ordre n	Non	Numérique	Dépend de la méthode utilisée pour le calcul des coefficients	Stochastique

D’autres méthodes existent comme la méthode Dimension Reduction Method ou les polynômes de chaos [Rahman S., Xu H., 2004] [Wiener N., 1938] [Pichral L., 2013]. Le choix est porté sur la méthode PoV étant donné le rapport précision calcul qu’elle offre tout en s’adaptant aux distributions caractérisées par une moyenne et un écart-type [Pichral L., 2013]. De plus, la méthode Dimension Reduction Method présente dans le Tableau 4-1 demande de connaître les cinq premiers moments ordinaires pour chaque paramètre d’entrée ce qui n’est pas trivial.

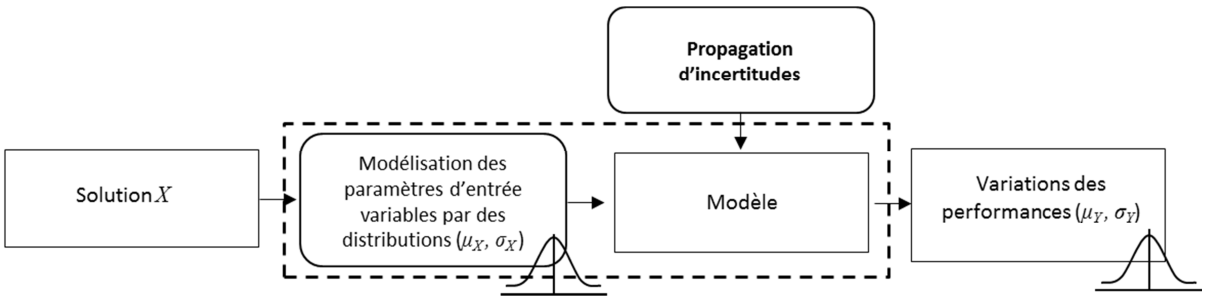

Figure 4-4. Propagation d’incertitudes

Dans cette partie, on considère un modèle de simulation entrées/sorties écrit sous la forme canonique suivante :

$$y = g(x)$$

avec $y \in \mathbb{R}^m$ le vecteur de sortie et $x \in \mathbb{R}^n$ celui des entrées. Les entrées sont caractérisées par leurs deux moments : la moyenne et l’écart-type. On chercher à calculer la moyenne et l’écart-type en sortie de modèle (voir la Figure 4-4). Les méthodes de propagation d’incertitudes supposent que les distributions des paramètres d’entrée soient connues.

A titre de comparaison, la méthode classique de propagation d’incertitudes par Monte-Carlo sera brièvement présentée. Ensuite, la méthode Propagation of Variance (PoV) sera présentée et comparée à Monte-Carlo.

4.2.1. Monte-Carlo

La méthode Monte-Carlo est l'une des méthodes les plus utilisées. Ceci s'explique par sa facilité de mise en œuvre et ses résultats fiables. Elle est également intéressante car elle permet de calculer les performances sur un échantillon et donc permet une analyse statistique détaillée (calcul du taux de rebus, estimation des moments …) ce qui est très difficile à faire avec d'autres méthodes. La méthode consiste à faire un échantillon sur les paramètres d'entrée suivant leurs densités de probabilité et d'évaluer les sorties pour chaque point de l'échantillon [Amelin, 2004]. Pour un échantillon de taille N, des vecteurs $x^i(i = 1, \dots, N)$ sont tirés suivant les lois de distribution de chacun des paramètres d'entrée. Pour chaque vecteur d'entrée les sorties $y^i = g(x^i)(i = 1, \dots, N)$ sont évaluées. Ensuite, les moments des sorties sont calculés via les estimateurs suivants :

$$\mu_y \approx \frac{1}{N}\sum_{i=1}^{N} g(x^i) \tag{4.11}$$

$${\sigma_y}^2 \approx \frac{1}{N}\sum_{i=1}^{N} \left(g(x^i) - \mu_y\right)^2 \tag{4.12}$$

La méthode de propagation d'incertitudes par Monte-Carlo est une méthode non-intrusive car elle ne nécessite que l'évaluation du modèle sur un échantillon défini. Elle convient donc aussi bien aux modèles analytiques qu'aux modèles boites noires. C'est une méthode très facile à implémenter et à utiliser et qui donne des résultats très fiables pourvu que la taille de l'échantillon le permette. La précision de cette méthode dépend de la taille de l'échantillon utilisé qui elle-même dépend du degré de non linéarité du modèle, des paramètres d'entrées etc … En effet, plus les dispersions des paramètres d'entrée sont importantes, plus il est nécessaire d'avoir un échantillon grand afin de couvrir leurs variabilités. Certains auteurs donnent des informations sur le choix de la taille de l'échantillon de l'ordre de 10^4 à 10^6 afin d'obtenir des résultats fiables [Lemaire, 2005].

Cette méthode est adaptée pour analyser la robustesse d'une solution de conception. Elle est cependant inappropriée pour l'optimisation robuste qui nécessite une méthode intégrée dans la boucle d'optimisation et qui sera donc appelée un grand nombre de fois.

D'autres variantes de Monte-Carlo existent et c'est principalement des améliorations de l'échantillonnage. Ceci peut être réalisé en utilisant un échantillonnage préférentiel comme dans [Lu and Zhang, 2003] ou la méthode du Latin-Hypercube [Olsson et al., 2003]. Il est également possible d'utiliser des modèles de substitution ou méta-modèles. Cette technique consiste à construire une surface de réponse du modèle initial de façon à avoir un modèle grossier mais léger en calcul. Plusieurs méthodes existent pour construire ce modèle de substitution comme la régression polynomiale [Jin et al., 2000] [Vivier, 2002], le Krigeage (Kriging) [Laurenceau, 2008], [Jin et al., 2000] ou encore Radial Basis Function [Forrester and Keane, 2009].

4.2.2. Propagation of Variance : PoV

La méthode Propagation of Variance (abréviation PoV) utilise le développement de Taylor localement pour calculer les moments [Lee and Park, 2001] [Glancy, 1999]. Le modèle $y = g(x)$ peut être approximé par la série de Taylor autour d'un point μ_x:

$$y = g(\mu_x) + \sum_{k=1}^{r} \frac{g^{(k)}(\mu_x)}{k!}(x - \mu_x)^k + o(x^r) \tag{4.13}$$

Bien-entendu, g étant un modèle multidimensionnel, il faut par conséquent utiliser un développement de Taylor en plusieurs dimensions.

La méthode PoV au premier ordre suppose que le modèle est localement linéaire. Au second ordre, PoV suppose que le modèle est localement quadratique. Ceci dépend évidemment du degré de linéarité du modèle mais aussi de la valeur de σ. En effet, pour de faibles valeurs d'écart-type, ces suppositions sont vraies. Dès que la valeur de σ devient importante, il faut vérifier l'exactitude des approximations puisque son étendu est plus grande.

Pour mettre en évidence la dépendance entre l'écart-type et l'ordre du développement de Taylor, on prend l'exemple de la fonction $y = \cos(x)$ (Figure 4-5). On considère des valeurs d'écart-types exagérées par rapport à la réalité afin de mettre en exergue leurs effets. Nous avons utilisé Monte-Carlo sur les développements de Taylor afin de calculer les moyennes et écart-types en sortie du modèle approximé. Si on considère que x suit la loi normale $\mathcal{N}(0,1)$, Monte-Carlo sur le modèle exact donne $\mu_y = 0.606412$ et $\sigma_y = 0.447026$. Les résultats de la méthode Monte-Carlo pour différents ordres du développement de Taylor de la fonction cosinus sont résumés dans le

Tableau 4-2.

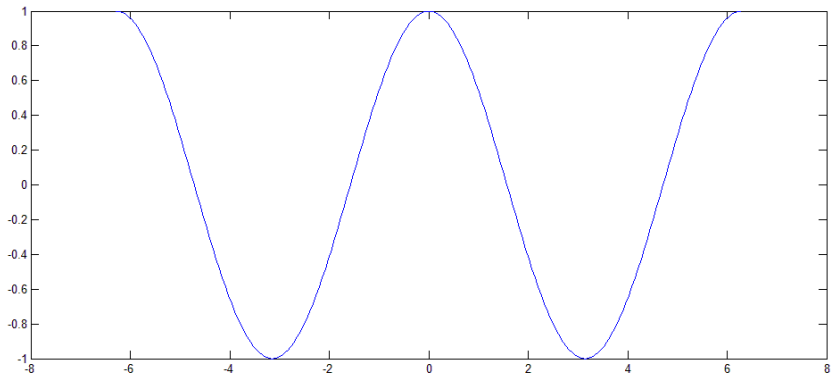

Figure 4-5. Fonction cosinus dans l'intervalle $[-2\pi, 2\pi]$

Tableau 4-2. Propagation de variance de la fonction cosinus : $\mathcal{N}(0,1)$

Ordre	Moyenne		Ecart-type	
	Valeur	Erreur	Valeur	Erreur
1	1	65%	0	100%
3	0.50028	17.5%	0.70602	58%
5	0.62478	3.03%	0.40798	8.76%
7	0.60408	0.38%	0.45491	1.76%
8	0.60664	0.04%	0.44631	0.16%

D'après le

Tableau 4-2, on voit que plus on monte dans l'ordre du développement de Taylor, plus on se rapproche des valeurs exactes. Dans le cas de $\sigma_x = 1$, il faut monter à l'ordre 8 pour avoir des résultats précis.

Maintenant, si on augmente encore la valeur de σ_x à 1.5, on passe donc à une loi normale $\mathcal{N}(0,1.5)$. Les valeurs données par un Monte-Carlo sur le modèle exacte sont $\mu_y = 0.324367$ et $\sigma_y = 0.632942$. Le Tableau 4-3 résume les résultats obtenus pour différents ordres du développement de Taylor.

Tableau 4-3. Propagation de variance de la fonction cosinus : $\mathcal{N}(0,1.5)$

Ordre	Moyenne		Ecart-type	
	Valeur	Erreur	Valeur	Erreur
1	1	208%	0	100%
3	-0.12493	138%	1.58855	150%
5	0.50656	56%	1.03633	63.7%
7	0.27094	16.45%	0.89315	41.12%
8	0.33661	3.8%	0.67035	5.9%
10	0.32208	0.68%	0.6371	0.66%
12	0.32472	0.13%	0.63324	0.05%

D'après le Tableau 4-3, il apparait qu'il faut monter à l'ordre 12 pour avoir des résultats comparables à ceux de Monte-Carlo sur le modèle exacte. Donc, plus la valeur de σ augmente et plus il faut monter en ordre de Taylor.

Le choix de la fonction $cosinus$ n'est pas arbitraire et a été fait intentionnellement. En effet, la fonction $cosinus$ est très particulière puisqu'elle a un degré de non-linéarités « infini » : plus on augmente l'intervalle de travail, et plus il faut un développement de Taylor d'ordre élevé pour approximer la fonction. Une astuce assez simple et fiable consiste à prendre l'intervalle $[\mu_x - 3\sigma_x; \mu_x + 3\sigma_x]$ qui contient 99,73% de l'échantillon en supposant les lois normales.

Les ordres de grandeurs de l'écart-type utilisés (1 puis 1.5) sont très grands par rapport aux ordres de grandeurs qu'on trouve en dimensionnement. C'est pour cette raison qu'il a fallu utiliser des développements de Taylor à des ordres élevés. L'objectif était de montrer l'impact de l'écart-type sur la précision du développement de Taylor. Ainsi, les forts écart-types posent un problème à la méthode PoV.

Dans la pratique, les écart-types en développement de produit sont beaucoup plus faibles. Les ordres 1 et 2 du développement de Taylor sont souvent suffisants en conception de produit. [Lee and Park, 2001] suggèrent d'utiliser PoV au premier ordre en indiquant que c'est valable pour des modèles à faibles non-linéarités. [Glancy, 1999] montre que PoV au premier ordre donne des résultats équivalents à Monte-Carlo avec *30000* tirages tandis que PoV au second ordre s'apparente à Monte-Carlo avec 10^6 tirages.

Pour une fonction g non linéaire, ce qui est généralement le cas, on propose d'utiliser d'une approximation quadratique donnée par le développement de Taylor au second ordre autour de la valeur nominale μ_x :

$$y = g(\mu_x) \approx g(\mu_x) + \sum_{i=1}^{n} \frac{\partial g(\mu_x)}{\partial x_i}\left(x_i - \mu_{x_i}\right) + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g(\mu_x)}{\partial x_i \partial x_j}\left(x_i - \mu_{x_i}\right)\left(x_j - \mu_{x_j}\right) \quad (4.14)$$

Souvent on fait la simplification d'estimer la moyenne des sorties avec $\mu_y \approx g(\mu_x)$. Ceci n'est correct que si g est localement linéaire. Le $k^{ième}$ moment statistique s'exprime via l'équation suivante :

$$E[y^k] = E\left[\left(g(x)\right)^k\right] = \int_{\mathbb{R}^n} \left(g(x)\right)^k f_x(x) dx_1 \dots dx_n \quad (4.15)$$

Par conséquent :

$$E[y^k] = \int_{\underline{x_1}}^{\overline{x_1}} \dots \int_{\underline{x_n}}^{\overline{x_n}} \left(g(x_1, \dots, x_n)\right)^k f_{x_1,\dots,x_n}(x_1, \dots, x_n) dx_1 \dots dx_n \quad (4.16)$$

Ainsi, pour calculer la moyenne en sortie :

$$\mu_y = E[y] = E[g(x)] \quad (4.17)$$

Donc pour un développement de Taylor au second ordre :

$$\mu_y \approx g(\mu_x) + \sum_{i=1}^{n} \frac{\partial g(\mu_x)}{\partial x_i} E\left[\left(x_i - \mu_{x_i}\right)\right] + \frac{1}{2!} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 g(\mu_x)}{\partial x_i \partial x_j} E\left[\left(x_i - \mu_{x_i}\right)\left(x_j - \mu_{x_j}\right)\right] \quad (4.18)$$

or,

$$E\left[\left(x_i - \mu_{x_i}\right)\right] = 0$$

$$E\left[\left(x_i - \mu_{x_i}\right)\left(x_j - \mu_{x_j}\right) = 0 \; si \; i \neq j\right.$$

d'où la formule de la moyenne :

$$\mu_y \approx g(\mu_x) + \frac{1}{2} \sum_{i=1}^{n} \frac{\partial^2 g}{\partial {x_i}^2} (\mu_x) \, \sigma_{x_i}^2 \quad (4.19)$$

On note que pour le développement de Taylor d'ordre 2, l'écart-type influence la moyenne de la sortie, d'où la distinction entre la valeur nominale et la valeur moyenne.

Pour le calcul de l'écart-type, un raisonnement similaire est adopté :

$$\sigma_y^2 = E\left[\left(y - \mu_y\right)^2\right] = E[(g(x) - \mu_y)^2]$$

Ainsi on obtient la formule de l'écart-type avec un développement de Taylor au second ordre :

$$\sigma_y^2 \approx \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i} (\mu_x) \right)^2 \sigma_{x_i}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{\partial^2 g}{\partial x_i \partial x_j} (\mu_x) \right)^2 \sigma_{x_i}^2 \sigma_{x_j}^2 \quad (4.20)$$

La reformulation robuste du modèle d'optimisation donne donc :

$$g_r(\mu_X, \sigma_X): \begin{cases} \mu_y \approx g(\mu_x) + \frac{1}{2} \sum_{i=1}^{n} \frac{\partial^2 g}{\partial {x_i}^2} (\mu_x) \, \sigma_{x_i}^2 \\ \sigma_y^2 \approx \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i} (\mu_x) \right)^2 \sigma_{x_i}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{\partial^2 g}{\partial x_i \partial x_j} (\mu_x) \right)^2 \sigma_{x_i}^2 \sigma_{x_j}^2 \end{cases} \quad (4.21)$$

Cette approximation permet de calculer les moments des paramètres de sortie en fonction des moments des paramètres en entrée. Elle est fondée sur un développement de Taylor au second ordre.

Dans certains cas où on manque d'informations sur la nature du modèle (non linéarités par exemple), il est conseillé de valider la fiabilité de la méthode PoV sur le modèle en question et avec les variabilités données en entrée avant de passer à l'optimisation. Ceci se fait en comparant les résultats obtenus par PoV et par Monte-Carlo et en s'assurant que les écarts sont tolérables. Dans les exemples de tests numériques, cette étape est systématiquement appliquée afin de montrer la validité de PoV.

PoV demande $2n^2+3n$ appels au modèle avec n la dimension du modèle soit le nombre de paramètres d'entrée. PoV constitue par conséquent une bonne alternative à Monte-Carlo et représente un bon compromis entre la précision et la rapidité. Ces deux critères sont très importants d'autant plus que la propagation d'incertitudes sera intégrée dans un algorithme d'optimisation.

La méthodologie PoV présentée dans ce paragraphe est valable pour des modèles statiques sans fonctionnelles. Etant donné l'importance des fonctionnelles en modélisation de produit, il serait intéressant d'étendre cette méthode aux modèles avec fonctionnelles. Dans ce qui suit, une extension aux modèles dynamiques intégrant des contraintes du type équations différentielles sera présentée.

4.2.3. Optimisation robuste du moteur à aimants permanents

Tout d'abord, un test de validation permettant de vérifier l'exactitude des calculs obtenus par la méthode PoV est réalisé. Ce test consiste en la comparaison entre les résultats obtenus par Monte-Carlo avec 10^6 tirages et la méthode PoV.

En utilisant la formulation introduite dans le chapitre sur IBBA, on fait varier tous les paramètres d'entrée avec un écart-type donné dans le Tableau 4-4.

Tableau 4-4. Lois des paramètres d'entrée du moteur

Paramètre	Moyenne	Ecart-type
β	0.8	0.01
E	0.003	0.0001
l_a	0.005	0.0001
e	0.001	0.0001

Les Ecart-types dans ce cas sont de l'ordre du 10% de la moyenne. Dans ce cas, les résultats des simulations avec PoV au second degré et Monte-Carlo sur le modèle exacte sont donnés dans le Tableau 4-5.

Tableau 4-5. PoV vs Monte-Carlo : moteur

	Monte-Carlo (10^6)		PoV		Erreur	
Paramètre de sortie	Moyenne	Ecart-type	Moyenne	Ecart-type	Moyenne	Ecart-type
B_e	0.4618	0.007567	0.4618	0.007561	0%	0.07%
C	0.006157	126.7 E-6	0.006157	126.8 E-6	0%	0.07%
D	0.1273	493 E-9	0.1273	0	0%	0.0003%
J_{cu}	6.15 E6	81538	6.15 E6	81392	0%	0.17%
K_f	0.1801	0.005793	0.1802	0.005788	0.05%	0.08%
λ	1.838	0.03271	1.838	0.0327	0%	0.03%
V_u	6.075 E-4	8.248 E-6	6.076 E-4	8.233 E-6	0.01%	0.18%

Ce tableau montre que les écarts entre PoV au second degré et Monte-Carlo sont négligeables : 0.05% d'écart maximal pour la moyenne et 0.18% d'écart maximal pour l'écart-type. Sachant que PoV demande uniquement ($2n^2$+3n) évaluations contre 10^6 pour Monte-Carlo, on considère que PoV est suffisamment précis.

Notons que dans le cas de la variable D, le pourcentage d'erreur de l'écart-type est mesuré par rapport à la moyenne vu que l'écart-type par PoV est nul.

Avant de commencer l'optimisation robuste, une optimisation de la moyenne ${\mu_{V_u}}^*$ et de l'écart-type ${\sigma_{V_u}}^*$ a été réalisée et ont donné ${\mu_{V_u}}^* = 6.0734\ e-4$ et ${\sigma_{V_u}}^* = 7.08\ e-6$.

Pour la fonction objectif suivante :

$$f_r = \alpha.\frac{\mu_{V_u}}{\mu_{V_u}{}^*} + (1-\alpha).\frac{\sigma_{V_u}}{\sigma_{V_u}{}^*} \tag{4.22}$$

des tests ont été réalisés pour différentes valeurs de α. Quelques solutions obtenues sont répertoriées dans le Tableau 4-6. Pour l'optimisation, CVI-PSO a été exécuté avec seulement 4 particules et 50 itérations.

Tableau 4-6. Optima pour différentes valeurs de α

α	μ_{V_u}	σ_{V_u}
0	6.47 E-4	7.080
0.2	6.45 E-4	7.082
0.4	6.39 E-4	7.123
0.6	6.23 E-4	7.3
0.8	6.12 E-4	7.61
1	6.07 E-4	8.23

Le front de Pareto est construit pour le problème bi-objectif $\min(\mu_{V_u}, \sigma_{V_u})$ et illustré par la Figure 4-6. Les techniques d'optimisation multi-objectifs permettent de générer un front de Pareto plus précis.

Etant donné que les solutions du front de Pareto sont non dominées, le concepteur doit choisir une de ses solutions. Dans le test numérique de l'actionneur différentiel, nous proposons un critère qui permet de classer ces solutions afin de faciliter le choix.

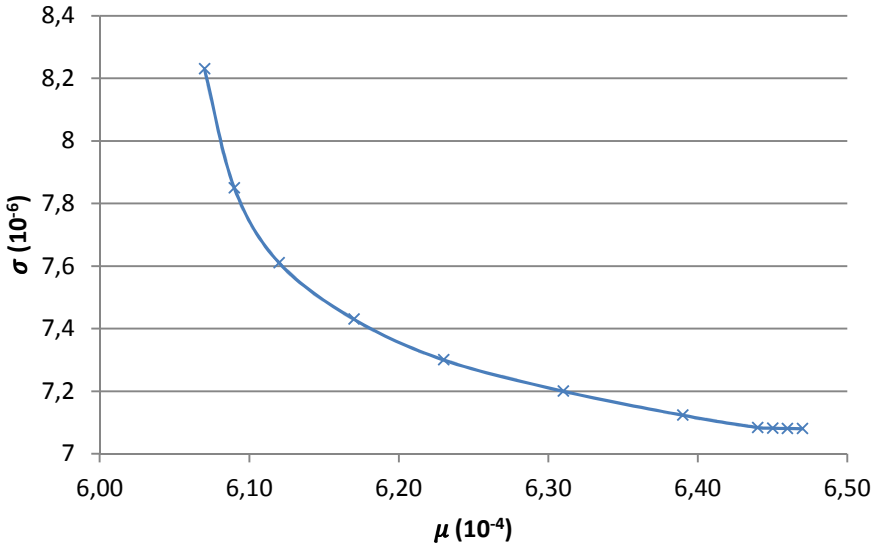

Figure 4-6. Optimisation robuste du moteur : Front de Pareto

4.3. Contribution : Propagation of Variance pour les modèles dynamiques

Un modèle d'optimisation dynamique est formulé comme suit :

$$\begin{cases} \min\limits_{x \in X \subseteq \mathbb{R}^n} \phi(\boldsymbol{u}(t), \boldsymbol{x}) \\ \quad \boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \\ \quad \boldsymbol{u}'(t) = \boldsymbol{h}(\boldsymbol{u}(t), \boldsymbol{x}) \\ \quad \boldsymbol{u}(t_0) = \boldsymbol{u_0}(\boldsymbol{x}) \\ \quad \boldsymbol{u}(\tau) \in \left[\underline{u}, \overline{u}\right]^{\tau}, \boldsymbol{x} \in \left[\underline{x}, \overline{x}\right], \boldsymbol{y} \in \left[\underline{y}, \overline{y}\right] \\ t \in \left[t_0, t_f\right], \qquad \tau \subset [t_0, t_f] \end{cases}$$

Ici, on s'intéresse à la propagation d'incertitudes sur la partie dynamique du modèle c'est-à-dire sur la partie $\boldsymbol{u}'(t) = \boldsymbol{h}(\boldsymbol{u}(t), \boldsymbol{x})$.

Dans le cas général, il n'est pas possible de résoudre analytiquement une équation différentielle même du premier ordre. Par exemple le système suivant de Lotka-Volterra :

$$\begin{cases} x' = ax - cxy \\ y' = -by + dxy \end{cases} \tag{4.23}$$

n'admet pas de solution exprimable analytiquement. Il est toutefois possible de décrire le comportement qualitatif des solutions. Une extension numérique de la méthode PoV est par conséquent nécessaire. Elle demande l'utilisation des dérivées numériques puisque les dérivées analytiques ne sont pas disponibles. La Figure 4-7 explique le fonctionnement de cette extension.

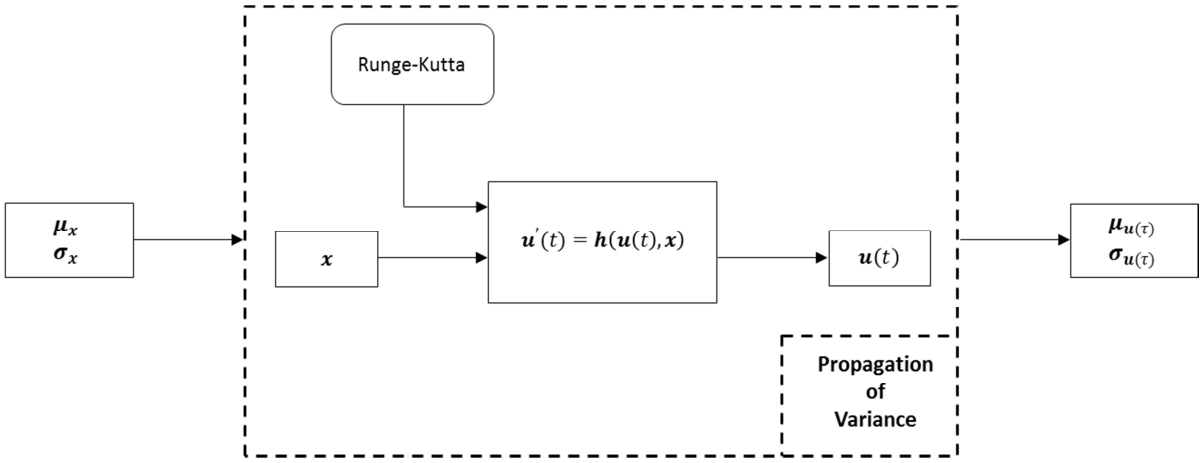

Figure 4-7. PoV sur le modèle dynamique

La propagation d'incertitudes est en effet appliquée à la partie dynamique du modèle (l'équation différentielle). Elle se greffe au-dessus du modèle dynamique résolu

numériquement (ici par Runge-Kutta 44). Runge-Kutta est utilisé afin de garder une cohérence avec l'algorithme d'optimisation CVI-PSO-RK44 introduit dans le chapitre 3.4. L'objectif étant de calculer la moyenne et l'écart-type en sortie de l'équation différentielle, c'est-à-dire $\mu_{\boldsymbol{u}(\tau)}$ et $\sigma_{\boldsymbol{u}(\tau)}$. Il est important de noter ici que les dérivées sont calculées par rapport aux paramètres de conception (les paramètres d'entrée) à chaque valeur de $t \in \tau$. Ainsi, l'équation différentielle ordinaire est considérée par la méthode PoV comme une boite noire.

$$\forall t \in \tau : \begin{cases} \mu_{\boldsymbol{u}(t)} \approx \boldsymbol{u}(t)_{\mu_{\boldsymbol{x}}} + \frac{1}{2}\sum_{i=1}^{n} \frac{\partial^2 \boldsymbol{u}(t)_{\boldsymbol{x}}}{\partial x_i^2}(\mu_x){\sigma_{x_i}}^2 \\ {\sigma_{\boldsymbol{u}(t)}}^2 \approx \sum_{i=1}^{n}\left(\frac{\partial \boldsymbol{u}(t)_{\boldsymbol{x}}}{\partial x_i}(\mu_x)\right)^2 \sigma_{x_i}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial^2 \boldsymbol{u}(t)_{\boldsymbol{x}}}{\partial x_i \partial x_j}(\mu_x)\right)^2 \sigma_{x_i}^2 \sigma_{x_j}^2 \end{cases} \tag{4.24}$$

où $\boldsymbol{u}(t)_{\boldsymbol{x}}$ la solution de l'équation différentielle $\boldsymbol{u}'(t) = \boldsymbol{h}(\boldsymbol{u}(t), \boldsymbol{x})$ à l'instant t.

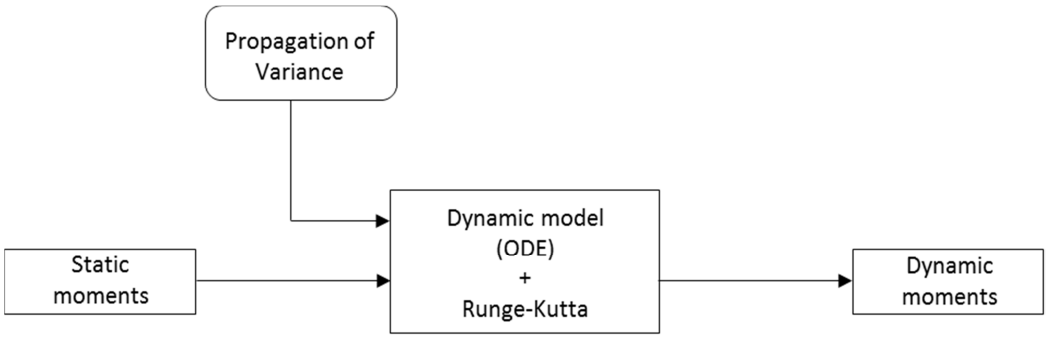

Figure 4-8. Procédure de calcul des Moments dynamiques

Comme indiqué par la Figure 4-8, PoV opère sur la solution du système d'équations différentielles pour chaque instant $t \in \tau$ en la dérivant par rapport aux variables d'entrée x_i. Concrètement, l'ODE sera résolue pour différents points x mais à un même instant t afin de calculer les dérivées.

Afin d'effectuer l'optimisation robuste sur un modèle dynamique, il est nécessaire de reformuler la fonction objectif et le cahier des charges. On adopte la formulation suivante :

$$\begin{cases} \min\limits_{x \in X \subseteq \mathbb{R}^n} \phi^r\left(\mu_{\boldsymbol{u}(t)}, \sigma_{\boldsymbol{u}(t)}, \mu_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}\right) \\ \left(\mu_{\boldsymbol{y}}, \sigma_{\boldsymbol{y}}\right) = G_r(\mu_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) \\ \left(\mu_{\boldsymbol{u}(t)}, \sigma_{\boldsymbol{u}(t)}\right) = \boldsymbol{h}_r(\boldsymbol{u}(t), \mu_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}) \\ \boldsymbol{u}(t_0) = \boldsymbol{u_0}(\boldsymbol{x}) \\ \left[\mu_{\boldsymbol{y}} - 3\sigma_{\boldsymbol{y}}; \mu_{\boldsymbol{y}} + 3\sigma_{\boldsymbol{y}}\right] \subseteq \left[\underline{y}, \overline{y}\right] \left[\mu_{\boldsymbol{x}} - 3\sigma_{\boldsymbol{x}}; \mu_{\boldsymbol{x}} + 3\sigma_{\boldsymbol{x}}\right] \subseteq \left[\underline{x}, \overline{x}\right] \\ \left[\mu_{\boldsymbol{u}(t)} - 3\sigma_{\boldsymbol{u}(t)}; \mu_{\boldsymbol{u}(t)} + 3\sigma_{\boldsymbol{u}(t)}\right] \subseteq \left[\underline{u}, \overline{u}\right] \\ t \in \left[t_0, t_f\right], \qquad \tau \subset [t_0, t_f] \end{cases} \tag{4.25}$$

Nous choisissons ici de nous limiter à l'étude de l'optimisation mono objectif. C'est pourquoi nous utilisons la fonction mono-objectif, ϕ^r écrite sous la forme :

$$\phi^r\left(\mu_{\boldsymbol{u}(t)}, \sigma_{\boldsymbol{u}(t)}, \mu_{\boldsymbol{x}}, \sigma_{\boldsymbol{x}}\right) = \alpha.\frac{\mu_\phi}{\mu_\phi{}^*} + (1-\alpha).\frac{\sigma_\phi}{\sigma_\phi{}^*} \tag{4.26}$$

où μ_ϕ et σ_ϕ sont respectivement la moyenne et l'écart-type de la fonction $\phi(\boldsymbol{u}(t), \boldsymbol{x})$. $\mu_\phi{}^*$ et $\sigma_\phi{}^*$ sont les valeurs obtenues en minimisant respectivement μ_ϕ et σ_ϕ.

4.4. Tests numériques : optimisation robuste de l'actionneur différentiel

Des tests numériques sur l'optimisation robuste sont effectués sur le modèle de l'actionneur différentiel (dynamique). On dispose d'un certain nombre d'outils qui permettent de faire la propagation d'incertitudes avec les méthodes PoV et Monte-Carlo [Pro@DESIGN]. Cet outil permet de faire le calcul des Moments en sortie d'un modèle. Un travail d'intégration entre Pro@DESIGN et des algorithmes d'optimisation développés en Matlab et Java a été fait afin de faire les tests numériques.

Les mêmes tests de validations que pour le moteur ont été effectués sur le modèle de l'actionneur différentiel afin de comparer PoV et Monte-Carlo. Les résultats sont donnés dans le Tableau 4-7. Dans ce tableau, une liste non-exhaustive de variables de sortie est considérée. L'erreur maximale est de l'ordre du 0.17% ce qui est très faible.

Tableau 4-7. PoV vs Monte-Carlo : actionneur

Paramètre de sortie		Monte-Carlo (10^6)		PoV		Erreur	
		Moyenne	Ecart-type	Moyenne	Ecart-type	Moyenne	Ecart-type
Statique	$B_{noystat}$	0.3085	0.00511	0.3085	0.00511	0.02%	0.04%
	C_1	1 E-4	1.42 E-5	1 E-4	1.41 E-5	0.01%	0.06%
	C_2	9.7 E-5	1.02 E-4	9.7 E-5	1.01 E-4	0.04%	0.01%
	C_3	1 E-6	5.30 E-7	0.999 E-6	5.31 E-7	0.08%	0.17%
	D_{ens}	2 E+8	658218	2 E+8	658304	0%	0.01%
	V_{max}	100	1.06	100.01	1.061	0.01%	0.1%
	$Volume$	1.46 E-6	9.85 E-9	1.46 E-6	9.84 E-9	0.01%	0.13%
Dynamique	Z_{tferm}	0.002	0	0.002	1.29 E-8	0%	0%

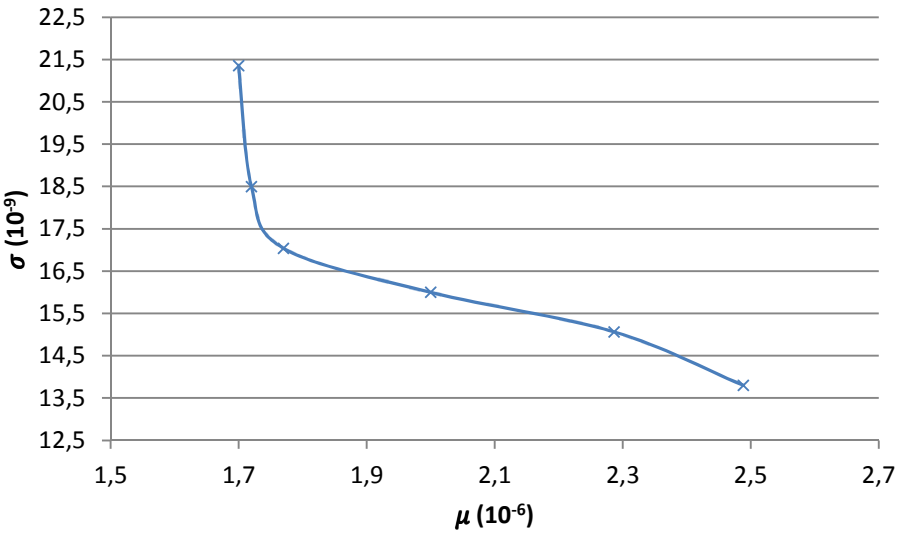

Figure 4-9. Optimisation robuste de l'actionneur différentiel : Front de Pareto

De la même façon que pour le moteur à aimants permanents, un front de Pareto a été construit dans la Figure 4-9. Ce front de Pareto est constitué de l'ensemble des solutions non dominées du problème d'optimisation bi-objectif $min(\mu_{V_u}, \sigma_{V_u})$. Ces solutions ne peuvent pas être classées entre elles d'où la difficulté de choisir une solution parmi d'autres. Afin d'aider le concepteur à choisir une solution, un indicateur de performance a été introduit. Si on part de la solution non robuste ($\alpha = 1$), chaque solution du front de Pareto sera meilleure du point de vue de l'écart-type et pire du point de vu de la fonction objectif. L'indicateur mesure le rapport entre le gain obtenu en écart-type et la perte en valeur moyenne.

Sachant que la solution non robuste a une moyenne $\mu_{Ref} = 1.7\ 10^{-6}$ et un écart-type $\sigma_{Ref} = 21.36\ 10^{-9}$, le gain de la solution $(\mu_{Sol}, \sigma_{Sol})$ est donné par la formule :

$$Index_{Sol} = \frac{\sigma_{Sol}^{gain}}{\mu_{Sol}^{perte}} \tag{4.27}$$

avec :

$$\begin{cases} \sigma_{Sol}^{gain} = \dfrac{\sigma_{Ref} - \sigma_{Sol}}{\sigma_{Ref}} \\ \mu_{Sol}^{perte} = \dfrac{\mu_{Sol} - \mu_{Ref}}{\mu_{Ref}} \end{cases} \tag{4.28}$$

Cet indice permet de tracer l'histogramme de la Figure 4-10 pour 8 solutions particulières du front de Pareto [Picheral L., Mazhoud I. et al, 2013].

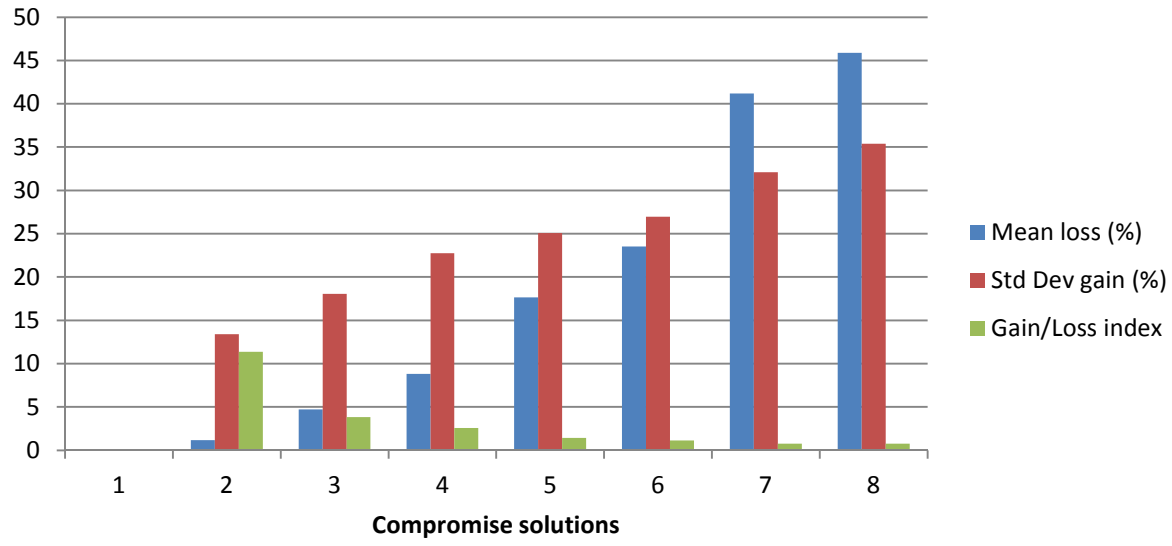

Figure 4-10. Indicateur de performance des solutions de compromis

La Figure 4-10 montre l’intérêt de la solution numéro 2. En effet, cette solution permet de gagner en robustesse tout en ayant une faible perte au niveau de sa moyenne en comparaison avec la solution non robuste. C’est l’idée de cet indicateur qui mesure la balance entre le gain en robustesse et la perte en moyenne. La solution 2 présente donc le meilleur compromis.

4.5. Conclusion

Ce chapitre a introduit l'aspect robustesse dans l'optimisation. Ceci peut se faire grâce à l'utilisation d'une méthode de propagation d'incertitudes, en l'occurrence PoV. La méthode PoV a été choisie pour le compromis qu'elle propose entre la précision et la rapidité.

Une extension de PoV aux modèles avec des contraintes du type équations différentielles a été proposée. L'optimisation robuste se fait donc en combinant PoV avec l'algorithme CVI-PSO développé dans le chapitre précédent.

Egalement, un indicateur a été introduit afin d'orienter l'ingénieur dans le choix de la solution la plus adaptée parmi les solutions du front de Pareto. Cet indicateur mesure le rapport entre le gain en robustesse et la perte en moyenne par rapport à la solution optimale non robuste.

Ces travaux ont fait l'objet d'une participation à la conférence internationale IEEE Compumag [Picheral L., Mazhoud I. et al, 2013].

5. Conclusion et perspectives

Dans les chapitres de ce mémoire, nous avons proposé des outils d'optimisation adaptés à différentes conditions. La progression dans les chapitres a suivi la difficulté des modèles de pré-dimensionnement. Cette difficulté provient de trois paramètres principaux :

- la taille des modèles
- la nature des équations
- la prise en compte des variabilités

Plus la taille du modèle est importante, plus le temps d'évaluation est important. Ceci ajoute une contrainte non négligeable sur le choix des méthodes d'optimisation les plus adaptées. C'est ce qui explique l'utilisation de l'algorithme déterministe à base de calcul d'intervalles IBBA. En effet, il permet de proposer des solutions optimales exactes mais avec une complexité exponentielle par rapport au nombre de paramètres du modèle. Une reformulation a été introduite et a permis de réduire considérablement le nombre d'itérations tout en améliorant la précision de résolution. Cela dit, cet algorithme reste limité car la reformulation n'a fait que pousser ses limites un peu plus loin. De plus, l'utilisation d'IBBA nécessite une reformulation et donc une requalification du modèle.

Afin de s'affranchir de la contrainte liée à la taille des modèles tout en gardant des performances convenables, nous avons proposé un algorithme évolutionnaire à base d'essaims particulaires (PSO) avec un nouveau mécanisme de gestion de contraintes : CVI-PSO. Ce mécanisme fondé sur un critère lexicographique combiné à une normalisation des violation via l'arithmétique d'intervalles a permis d'avoir des performances très intéressantes validées par la comparaison avec plusieurs autres algorithmes existants.

Ensuite, nous avons traité les problèmes d'optimisation avec des contraintes de nature autre qu'algébriques. Dans ce travail, nous nous somme focalisé sur les contraintes du type équations différentielles (ODE). Nous avons proposé une extension de l'algorithme CVI-PSO. Cette extension combine l'algorithme CVI-PSO avec un algorithme de résolution d'équations différentielles à base de Runge-Kutta. L'algorithme obtenu est nommé PSO-RK44.

Enfin, nous avons voulu intégrer les variabilités dans le processus d'optimisation. Ceci nécessite la reformulation du modèle afin de faire la propagation d'incertitudes. La propagation d'incertitudes vise à calculer les variabilités des paramètres de sorties (performances) en fonction des variabilités sur les paramètres d'entrée (paramètres de conception). Afin de faire la propagation d'incertitudes, nous avons utilisé la méthode PoV qui permet un bon compromis entre la précision et le temps de calcul. Nous avons proposé une extension de la méthode PoV afin de prendre en considération les modèles dynamiques incluant des contraintes du type équations différentielles.

Ces contributions ont fait l'objet d'articles de journaux et de participation à des conférences nationales et internationales [Mazhoud I. et al, 2011a] [Mazhoud I. et al, 2011b] [Mazhoud I.

et al, 2012a] [Mazhoud I. et al, 2012b] [Mazhoud I. et al, 2012c] [Mazhoud I. et al, 2012d] [Mazhoud I. et al, 2013] [Picheral L., Mazhoud I. et al, 2013].

Ce qui peut être réalisé dans la suite de ce travail peut se résumer dans différents points :

- Extension de l'algorithme IBBA avec la reformulation aux modèles de faible dimension avec des contraintes du type équations différentielles. En effet, des méthodes de résolution exacte d'ODE existent dans la littérature et dans des bibliothèques comme VNODE-LP [Nedialkov N. S. et al, 1999] [Nedialkov N. S., 2006]

- Développer des outils adaptés à l'optimisation de modèles de plus grande dimension. Ce demande par exemple l'utilisation des surfaces de réponse qui permet de créer un modèle de substitution au modèles initial [Jin et al., 2000] [Vivier, 2002] [Laurenceau, 2008] [Jin et al., 2000] [Forrester and Keane, 2009]. Ce modèle est moins précis mais beaucoup plus rapide à calculer ce qui permet d'utiliser un large panel de méthodes d'optimisation efficaces tels que les méthodes stochastiques. Il est également possible d'utiliser des méthodes de space mapping qui fait l'optimisation sur un modèle grossier (surface de réponse par exemple) tout en gardant un lien avec le modèle précis [Hage-Hassan M., Remy G. et al, 2012] [Tran T. V., Brisset S., Brochet P., 2009]. Cette méthode permet d'affiner la précision du modèle grossier au fur et à mesure de l'optimisation. Ceci permettra de traiter des modèles issus de logiciels de simulation tels que les éléments finis ou les modèles à base de circuit.

- Développer la robustesse et la propagation d'incertitudes aux modèles de plus grande dimension, en lien avec le point précédent.

6. Références

Aguirre, A.H., Rionda, S.B., Coello, C.C.A., Lizaraga, G.L., Montes, M.E., 2004. Handling constraints using multi-objective optimization concepts. Int. J. Numer. Methods Eng. 59 (15), 1989–2017.

Amelin, M., 2004. On Monte Carlo simulation and analysis of electricity markets. dissertation. KTH.

Aragon, V.S., Esquivel, S.C., Coello, C.A.C., 2010. A modified version of a T-Cell Algorithm for constrained optimization problems. Int. J. Numer. Methods Eng. 84 (3), 351–378.

Araya, I., Trombettoni, G., et al., 2009. An Interval Constraint Propagation Algorithm Exploiting Monotonicity. pp. 65-83.

Arumugam, M., Rao, M., Chandramohan, A., 2008. A new and improved version of particle swarm optimization algorithm with global-local best parameters. Knowl. Inf. Syst. 16 (3), 324–350.

Baumann, E., 1988. Optimal Centered Forms. BIT Numerical Mathematics, v. 28, pp. 80-87.

Belegundu, A.D., 1982. A Study of Mathematical Programming Methods for Structural Optimization. University of Iowa.

Berliner, C., Brimson, J. A., 1988. Cost management for today's advanced manufacturing. Harvard Business School Press.

Bird, J. O. Engineering Mathematics. Newnes, 2007.

Campana, E.F., Fasano, G., Pinto, A., 2010. Dynamic analysis for the selection of parameters and initial population, in particle swarm optimization. J. Glob. Opt. 48 (3), pp. 347–397.

Chabert, G., Jaulin, L., 2009. Hull-Consistency Under Monotonicity. Principles and Practice of Constraint Programming. Springer Berlin Heidelberg.

Coello, C.A.C., 2000. Use of a self-adaptive penalty approach for engineering optimization problems. Comput. Ind. 41, pp. 113–127.

Darnault, R., Bigeon, J., 2005. Etude de cas : dispositif de déclenchement. Design Processing Technologies. Rapport interne.

Deb, K., 1991. Optimal design of a welded beam via genetic algorithms. AIAA J. 29 (11), pp. 2013–2015.

Du, X., Venigella, P.K., Liu, D., 2009. Robust mechanism synthesis with random and interval variables. Mechanism and Machine Theory *44*, pp. 1321–1337.

Faltings, B. et al., 2001. Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art. COCONUT project.

Forrester, A.I.J., and Keane, A.J., 2009. Recent advances in surrogate-based optimization. Progress in Aerospace Sciences 45, pp. 50–79.

Glancy, C.G., 1999. A Second-Order Method for Assembly Tolerance Analysis. In In Proceedings of the ASME Design Automation Conference.

Hage-Hassan, M., Remy, G., Krebs, G., Marchand, C., 2012. Radial output space mapping for electromechanical systems design. OIPE 2012, Belgique.

Hansen, E., 1979. Global Optimization Using Interval Analysis - The One-Dimensional Case. Journal of Optimization Theory and Application, v. 23, pp. 331-344.

Hansen, E., 1980. Global Optimization Using Interval Analysis - The Multi-Dimensional Case. Numerische Mathematik, v. 34, pp. 247-270.

Hansen, E., Walster, G. W., 2004. Global Optimization Using Interval Analysis. Edition 2: Marcel Dekker,.

He, Q., Wang, L., 2007a. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. 20 (1), 89–99.

He, Q., Wang, L., 2007b. A hybrid particle swarm optimization with a feasibility- based rule for constrained optimization. Appl. Math. Comput. 186 (2), 1407–1422.

He, S., Prempain, E., Wu, Q.H., 2004. An improved particle swarm optimizer for mechanical design optimization problems. Eng. Opt. 36 (5), 585–605.

Hu, X.H., Eberhart, R.C., 2002. Solving constrained nonlinear optimization pro- blems with particle swarm optimization. In: World Multiconference on Systematics, Cybernetics and Informatics.

Janson, S., Merkle, D., Middendorf, M., 2008. Molecular docking with multi- objective Particle Swarm Optimization. Appl. Soft Comput. 8 (1), 666–675.

Benhamou, F., Goualard, F., Granvilliers, L., Puget, J. F., 1999. Revising Hull and Box Consistency. Int. Conf. on Logic Programming. MIT Press. pp. 230-244.

Jin, R., Chen, W., and Simpson, T.W., 2000. Comparative Studies Of Metamodeling Techniques Under Multiple Modeling Criteria. Structural and Multidisciplinary Optimization 23, 1–13.

Kannan, B.K., Kramer, S.N., 1994. An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J. Mech. Des. 116, 318–320.

Kearfott, R. B., 1991. An interval branch and bound algorithm for bound constrained optimization problems. Journal of Global Optimization, pp. 259-280.

Kearfott, R. B., 1996. Interval Computations: Introduction, Uses and Ressources. Euromath Bulletin, v. 2, pp. 95-112.

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. In: IEEE Interna- tional Conference on Neural Networks, vol. 4. IEEE Press, New York, pp. 1942– 1948.

Kone, A.D., Nogarede, B., Lajoie Mazenc, M.,1993. Le dimensionnement des actionneurs électriques : un problème de programmation non linéaire. J. Phys. 3,285–301.

Laurenceau, J., 2008. Surfaces de réponse par krigeage pour l'optimisation de formes aérodynamiques. PhD Thesis.

Lee K., Park G., 2001. Robust optimization considering tolerances of design variables. Computers and structures, vol.79, no.1, pp.77-86.

Lee, I., Choi, K.K., Du, L., Gorsich, D. 2008. Dimension reduction method for reliability-based robust design optimization. Computers & Structures *86*, pp. 1550–1562.

Lemaire, M., 2005. Fiabilité des structures. Hermes Science Publications.

Liu, X., 2007. Parameterized defuzzification with maximum entropy weighting function—Another view of the weighting function expectation method. Mathematical and Computer Modelling *45*, pp. 177–188.

Liu, H., Cai, Z., Wang, Y., 2010. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl. Soft Comput. 10 (2), 629–640.

Lu, Z., and Zhang, D., 2003. On importance sampling Monte Carlo approach to uncertainty analysis for flow and transport in porous media. Advances in Water Resources 26, 1177–1188.

Lu, H.Y., Chen, W.Q., 2006. Dynamic-objective particle swarm optimization for constrained optimization problems. J. Comb. Opt. 12 (4), 409–419.

Lu, H.Y., Chen, W.Q., 2008. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. J. Glob. Opt. 41 (3), 427–445.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Joyeux, P., 2013. Particle Swarm Optimization for solving engineering problems: a new constraint-handling mechanism. Engineering Applications of Artificial Intelligence. vol.26. no.4. pp.1263-1273.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Remy, G., 2012a. Interval-based global optimization in engineering using model reformulation and constraint propagation. Engineering Applications of Artificial Intelligence. vol.25. no.2. pp.404-417.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Remy, G., 2012b. The electromagnetic actuator design problem: An adapted interval global optimization algorithm. IEEE Transaction on Magnetics. vol.48. no.2. pp.387-390.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Joyeux, P., 2012c. Particle swarm optimization for dynamic analytical models involving ordinary differential equations. IEEE Xplore.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Joyeux, P., 2012d. Particle swarm optimization for dynamic analytical models involving ordinary differential equations. IEEE CCECE. Montréal.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., Remy, G., 2011a. The electromagnetic actuator design problem: An adapted interval global optimization algorithm using model reformulation and constraint propagation. IEEE Compumag. Sidney.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., 2011b. Interval global optimization using reformulation and contractor. ROADEF. Saint Etienne.

Messine, F., Nogarede, B., Lagouanelle, J.L., 1998. Optimal design of electromecha- nical actuators: a new method based on global optimization. IEEE Trans. Magn. 34 (1), 299–308.

Messine, F., 2004. Deterministic Global Optimization using Interval Constraint Propagation Techniques. RAIRO-OR, v. 38, pp. 277-294.

Mezura, M. E., Miranda-Varela, M.E., Gomez-Ramon, R.C., 2010. Differential evolu- tion in constrained numerical optimization: an empirical study. Inf. Sci. 180, pp. 4223–4262.

Michèle, A. H., 1983. Initiation aux techniques classiques de l'optimisation. Lavoisier,.

Montes, E.M., Coello, C.A.C., 2005. A simple multi-membered evolution strategy to solve constrained optimization problems. IEEE Trans. Evol. Comput. 9 (1), pp. 1–17.

Moore, R.E., 1966. Interval Analysis. Prentice-Hall, Englewood Cliffs.

Nedialkov, N. S., Jackson, K. R., Corliss, G. F., 1999. Validated Solutions of Initial Value Problems for Ordinary Differential Equations. Applied Mathematics and Computation, 105(1), pp. 21-68.

Nedialkov, N. S., 2006. Interval Tools for ODEs and DAEs. Proceedings of the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, IEEE Computer Society.

Olsson, A., Sandberg, G., and Dahlblom, O., 2003. On Latin hypercube sampling for structural reliability analysis. Structural Safety 25, 47–68.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K. H., 2007. Engineering Design: a systematic approach. Ed. 3. Springer.

Parsopoulos, K.E., Vrahatis, M.N., 2005. Unified particle swarm optimization for solving constrained engineering optimization problems. Lect. Notes Comput. Sci. 3612, pp. 582–591.

Picheral, L., 2013.Contribution à la conception préliminaire robuste en ingénierie de produit. PhD Thesis.

Picheral L., Mazhoud I., Hadj-Hamou, K., Bigeon, J., Joyeux, P., 2013. An automated optimization approach based on robust constraints and objective function. IEEE Compumag. Budapest.

Pulido, G.T., Coello, C.A.C., 2004. A constraint-handling mechanism for particle swarm optimization. In: IEEE Congress on Evolutionary Computation. IEEE Press, New York, pp. 1396–1403.

Pro@DESIGN. Design Processing Technologies. Site Web: www.designprocessing.com/

Rahman, S., Xu, H., 2004. A univariate dimension-reduction method for multi-dimensional integration in stochastic mechanics. Probabilistic Engineering Mechanics *19*, pp. 393–408.

Rao, S.S., 1996. Engineering Optimization. Wiley, New York.

Ren, Z., Pham, M. T., Song, M., Kim, D. H., Koh, C. S. 2011. A robust global optimization algorithm of electromagnetic devices utilizing gradient index and multi-objective optimization method. IEEE Trans. on Magn., vol.47, no.5, pp. 1254-1257.

Rolander, N., Rambo, J., Joshi, Y., Allen, J.K., and Mistree, F., 2006. An Approach to Robust Design of Turbulent Convective Systems. J. Mech. Des. *128*, pp. 844–855.

Rump, S.M.,1999. INTLAB - INTerval LABoratory. In:Csendes,T.(Ed.), Developments in Reliable Computing. Kluwer Academic Publishers, pp.77–104.

Scaravetti, D., 2004. Formalisation préalable d'un problème de conception, pour l'aide à la décision en conception préliminaire. Arts et Métiers ParisTech.

Sedlaczek, K., Eberhard, R.C., 2006. Using augmented Lagrangian particle swarm optimization for constrained problems in engineering. Struct. Multidiscip. Opt. 32 (4), pp. 277–286.

Shi, Y., Eberhart, R.C., 1998B. Parameter selection in particle swarm optimization. Evolutionary Programming VII, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 591–600.

Shi, Y., Eberhart, R., 1998A. A modified particle swarm optimizer. In: IEEE Congress on Evolutionary Computation. IEEE Press, New York, pp. 69–73.

Tamas, T. C., 1991. A new inclusion function for optimization: Kite -- the one-dimensional case. International Journal of Bifurcation and Chaos, v. 1, pp. 14-43.

Tran T. V., Brisset S., Brochet P., 2009. A New Efficient Method for Global Discrete Multi-Level Optimization combining Branch-and-Bound and Space-Mapping. IEEE Transactions on Magnetics, v. 45 n. 3.

Van Den Bergh, F., 2001. An Analysis of Particle Swarm Optimizers. PhD Thesis. Faculty of Natural and Agricultural Science, University of Pretoria.

Vaz, A.I.F., Vicente, L.N., 2007. A particle swarm pattern search method for bound constrained global optimization. J. Glob. Opt. 39 (2), pp. 197–219.

Vivier, S., 2002. Stratégies d'optimisation par la méthode des Plans d'Expériences, et Application aux dispositifs électrotechniques modélisés par Eléments Finis. Université des Sciences et Technologie de Lille - Lille I.

Voller, R. L., 1991. What Can Interval Analysis Do for Global Optimization? Journal of Global Optimization, v. 1, pp. 111-130.

Wang, J., Yin, Z., 2008. A ranking selection-based particle swarm optimizer for engineering design optimization problems. Struct. Multidiscip. Opt. 37 (2), pp. 131–147.

Wang, S., Liu, X., Qiu, J., Zhu, J. G., Guo, Y., Lin, Z. W., 2008. Robust Optimization in HTS Cable Based on DEPSO and Design for Six Sigma. Industry Applications Society Annual Meeting. IEEE, pp.1-5, 5-9.

Wang, Y., Cai, Z., Zhou, Y., Zeng, W., 2008. An adaptive trade-off model for constrained evolutionary optimization. IEEE Trans. Evol. Comput. 12 (1), pp. 80–92.

Wang, Y., Cai, Z., Zhou, Y., 2009. Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. Int. J. Numer. Methods Eng. 77 (11), pp. 1501–1534.

Wiener, N., 1938. The Homogeneous Chaos. American Journal of Mathematics *60*, pp. 897–936.

Worasucheep, C., 2008. A particle swarm optimization with stagnation detection and dispersion. In: IEEE Congress on Evolutionary Computation. IEEE Press, New York, pp. 424–429.

Zadeh, L. A., 1965. Fuzzy sets. Information and Control. 8, pp. 338-353.

7. Annexes

Annexe 1 : Benchmark de problèmes d’optimisation

Appendix: benchmarks and simulation results

Particle Swarm Optimization for solving engineering constrained problems

Issam Mazhoud · Khaled Hadj-Hamou · Jean Bigeon

The experiments are performed on well-known benchmarks including 24 test functions and 3 engineering problems. The characteristics of these problems are reported in the following Table, where:

- Var# is the number of variables
- Obj-f is the type of the objective function
- LInC# is the number of linear inequality constraints
- NLInC# is the number of non-linear inequalities
- LEC# is the number of linear equality constraints
- NLEC# is the number of non-linear equalities
- The coefficient $\alpha(\%)$ is defined experimentally by calculating the percentage of feasible solutions among 1 000 000 randomly generated solutions from the search space. This coefficient defines a measure of the difficulty of solving each problem.

K. Hadj-Hamou
Grenoble Institute of Technology - UJF - CNRS, Grenoble, France
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: khaled.hadj-hamou@grenoble-inp.fr

I. Mazhoud
Grenoble Institute of Technology - UJF - CNRS, Grenoble, France
E-mail: issam.mazhoud@grenoble-inp.fr

J. Bigeon
Grenoble Institute of Technology - UJF - CNRS, Grenoble, France
E-mail: jean.bigeon@grenoble-inp.fr

Table 1 Benchmark main characteristics: 24 test functions

Problem	Var#	Obj-f	LInC#	NLInC#	LEC#	NLEC#	α(%)
g01	13	quadratic	9	0	0	0	0.0001
g02	20	non-linear	1	1	0	0	99.996
g03	10	polynomial	0	0	0	1	0.0000
g04	5	quadratic	0	6	0	0	26.925
g05	4	cubic	2	0	0	3	0.0000
g06	2	cubic	0	2	0	0	0.0054
g07	10	quadratic	3	5	0	0	0.0002
g08	2	non-linear	0	2	0	0	0.8611
g09	7	polynomial	0	4	0	0	0.5314
g10	8	linear	3	3	0	0	0.0002
g11	2	quadratic	0	0	0	1	0.0000
g12	3	quadratic	0	1	0	0	4.5420
g13	5	non-linear	0	0	0	3	0.0000
g14	10	non-linear	0	0	3	0	0.0000
g15	3	quadratic	0	0	1	1	0.0000
g16	5	non-linear	4	34	0	0	0.0186
g17	6	non-linear	0	0	0	4	0.0000
g18	9	quadratic	0	13	0	0	0.0000
g19	15	non-linear	0	5	0	0	35.548
g20	24	linear	0	6	2	12	0.0000
g21	7	linear	0	1	0	5	0.0000
g22	22	linear	0	1	8	11	0.0000
g23	9	linear	0	2	3	1	0.0000
g24	2	linear	0	2	0	0	74.268
eng01	4	non-linear	3	1	0	0	39.737
eng02	4	non-linear	2	5	0	0	2.6638
eng03	3	non-linear	1	3	0	1	0.7560

Table 2 Statistical simulation results

Problem	Stat	CVI-PSO	AATM [2]	DECV [5]	T-CELL [4]
g01	best	−14.9999999999998	−15.000	−15.000	−15.0
−15.000	mean	−14.9999999999998	−15.000	−14.855	−15.0
	worst	−14.9999999999998	−15.000	−13.000	−15.0
	std. dev.	$0.45e-15$	$3.1e-07$	0.459	0.0
g02	best	−0.80009774225808	−0.803389	−0.704009	−0.801367
−0.803619	mean	−0.79087557689617	−0.791213	−0.569458	−0.752975
	worst	−0.74694209848138	−0.767040	−0.238203	−0.687827
	std. dev.	0.010912	0.0086	0.0951	0.032095
g03	best	−1.00000000000000	−1.00	−0.461	−1.0
−1.000	mean	−0.99999999999999	−1.00	−0.134	−1.0
	worst	−0.99999999999999	−1.00	−0.002	−1.0
	std. dev.	$0.37e-15$	$3.5e-04$	0.117	0.0
g04	best	−30665.8217102204	−30665.539	−30665.539	−30665.5385
−30665.539	mean	−30665.8209961102	−30665.539	−30665.539	−30665.5384
	worst	−30665.8032411366	−30665.539	−30665.539	−30665.5382
	std. dev.	0.003391	$1.0e-11$	$1.56e-06$	$1.0e-4$
g05	best	5127.27766734565	5126.498	5126.497	5126.6255
5126.497	mean	5127.27766734565	5126.714	5126.497	5378.2678
	worst	5127.27766734565	5128.824	5126.497	6112.1181
	std. dev.	0.0	0.43	0.0	298.0173
g06	best	−6961.81387558016	−6961.814	−6961.814	−6961.81387
−6961.814	mean	−6961.81387558016	−6961.814	−6961.814	−6961.81386
	worst	−6961.81387558016	−6961.814	−6961.814	−6961.81365
	std. dev.	0.0	$7.1e-12$	0.0	$3.9e-5$
g07	best	24.4738268493991	24.307	24.306	24.3209
24.306	mean	26.5612953717409	24.317	24.794	24.6534
	worst	29.5242543022246	24.356	29.511	25.1347
	std. dev.	1.642519	0.013	1.37	0.219815
g08	best	−0.10545950508841	−0.095825	−0.095825	−0.095825
−0.095825	mean	−0.10545950508841	−0.095825	−0.095825	−0.095825
	worst	−0.10545950508841	−0.095825	−0.095825	−0.095825
	std. dev.	0.0	$5.8e-18$	$4.23e-17$	0.0
g09	best	680.635400792284	680.630	680.630	680.63
680.630	mean	680.755705150086	680.634	680.630	680.65
	worst	680.863957829483	680.646	680.630	680.70
	std. dev.	0.079232	0.0045	$3.45e-07$	0.0167
g10	best	7049.27658551720	7049.603	7049.248	7050.8342
7049.248	mean	7053.21431059926	7077.477	7103.548	8020.7551
	worst	7091.88085911943	7183.295	7808.980	9054.2923
	std. dev.	10.615642	31	148	621.7231
g11	best	0.75000000000000	0.75	0.75	0.7499
0.75	mean	0.75000000000000	0.75	0.75	0.7499
	worst	0.75000000000000	0.75	0.75	0.7499
	std. dev.	0.0	$3.8e-06$	$1.12e-16$	0.0
g12	best	−1.00000000000000	−1.000	−1.000	−1.0
−1.000	mean	−1.00000000000000	−1.000	−1.000	−1.0
	worst	−1.00000000000000	−1.000	−1.000	−1.0
	std. dev.	0.0	0.0	0.0	0.0

Problem	Stat	CVI-PSO	AATM [2]	DECV [5]	T-CELL [4]
g13	best	0.05555582974209	NA	0.059798	0.054638
0.053942	mean	0.06559074439931	NA	0.382401	0.458857
	worst	0.09372815700166	NA	0.999094	0.994983
	std. dev.	0.010177	NA	0.268	0.344995
g14	best	−47.4530114308810	−47.762	−47.764888	−47.517100
−47.764888	mean	−44.4246904372610	−47.750	−47.722542	−45.310800
	worst	−42.4277271159680	−47.712	−47.036510	−43.272382
	std. dev.	1.406368	0.01	0.162	1.1156
g15	best	961.715707147147	961.715	961.715022	961.71502
961.715022	mean	961.718595474823	961.715	961.715022	963.37482
	worst	961.718781406392	961.716	961.715022	970.59467
	std. dev.	$6.87e-04$	$3.0e-04$	$2.31e-13$	2.27562
g16	best	−1.9051549999999	−1.905155	−1.905155	−1.905155
−1.905155	mean	−1.9051549999999	−1.905155	−1.905155	−1.905155
	worst	−1.9051549999998	−1.905155	−1.905149	−1.905155
	std. dev.	$8.52e-15$	$2.4e-14$	$1.10e-06$	0.0
g17	best	8853.53989132959	NA	8853.541289	8861.821
8853.539675	mean	8853.53989132959	NA	8919.936362	8990.997
	worst	8853.53989132959	NA	8938.571060	9231.201
	std. dev.	$3.70e-12$	NA	25.9	106.193174
g18	best	−0.86463128789989	−0.866025	−0.866025	−0.86596
−0.866025	mean	−0.80910925896692	−0.865952	−0.859657	−0.78455
	worst	−0.64443684728892	−0.864843	−0.674981	−0.62977
	std. dev.	0.062701	$2.1e-04$	0.0348	0.09746
g19	best	32.8270270899692	32.725	32.655593	33.39078
32.655593	mean	35.0673371976379	32.952	32.660587	38.92761
	worst	43.3749595963395	33.243	32.785360	48.48763
	std. dev.	2.286728	0.14	0.0237	3.19102
g20	best	0.21468967194629	NA	NA	NA
NA	mean	NA	NA	NA	NA
	worst	NA	NA	NA	NA
	std. dev.	NA	NA	NA	NA
g21	best	193.786925246823	NA	193.724510	NA
193.724510	mean	193.786935225931	NA	198.090578	NA
	worst	193.787093370887	NA	324.702842	NA
	std. dev.	$3.38e-05$	NA	23.9	NA
g22	best	NA	NA	NA	NA
236.430976	mean	NA	NA	NA	NA
	worst	NA	NA	NA	NA
	std. dev.	NA	NA	NA	NA
g23	best	−400.000000000000	NA	−400.055093	NA
−400.0551	mean	−400.000000000000	NA	−392.029610	NA
	worst	−400.000000000000	NA	−342.524522	NA
	std. dev.	0.0	NA	12.4	NA
g24	best	−5.50801327159532	−5.508013	−5.508013	−5.508013
−5.508013	mean	−5.50801327159532	−5.508013	−5.508013	−5.508013
	worst	−5.50801327159529	−5.508013	−5.508013	−5.508013
	std. dev.	$9.46e-15$	$1.8e-15$	$2.71e-15$	0.0

g01

$$\min f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$
$$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$
$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$
$$g_4(x) = -8x_1 + x_{10} \leq 0$$
$$g_5(x) = -8x_2 + x_{11} \leq 0$$
$$g_6(x) = -8x_3 + x_{12} \leq 0$$
$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0$$
$$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0$$
$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0$$
$$0 \leq x_i \leq 1, i = 1, \cdots, 9, 0 \leq x_i \leq 100, i = 10, \cdots, 12, 0 \leq x_{13} \leq 1$$

Simulation results obtained using CVI-PSO:

$best = -14.9999999999998$
$mean = -14.9999999999998$
$worst = -14.9999999999998$
$std.\ dev. = 0.45e-15$
$g_1 = -1.399982352268125e-10$ (active)
$g_2 = -1.537259208816977e-10$ (active)
$g_3 = -3.100808498857077e-11$ (active)
$g_4 = -5.000000000131358$
$g_5 = -5.000000000008640$
$g_6 = -5.000000000022368$
$g_7 = -1.313575914707599e-10$ (active)
$g_8 = -8.640199666842818e-12$ (active)
$g_9 = -2.236788532172795e-11$ (active)
$x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 2.999999999868642, 2.999999999991360, 2.999999999977632, 1)$

g02

$$\min f(x) = -\left| \frac{\sum_{i=1}^{n} \cos^4 x_i - 2\prod_{i=1}^{n} \cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i \cdot x_i^2}} \right|$$

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

$$n = 20, 0 \leq x_i \leq 10, i = 1, \cdots, n$$

Simulation results obtained using CVI-PSO:

$best = -0.8000977422580824$
$mean = -0.7908755768961707$
$worst = -0.7469420984813827$
$std.\ dev. = 1.091208615147532e-02$

$g_1 = -2.801348042424934e-11$ (active)
$g_2 = -119.8084204587099$
$x^* = (3.184632102285373, 3.136257454857324, 3.138073376063323,$
$3.113218987291479, 3.043076226876065, 3.021690028942750,$
$3.000000000000000, 3.000013894078654, 0.534611799406777,$
$0.447814951197522, 0.449038043845460, 0.369326814571331,$
$0.464794140190711, 0.472311056088807, 0.485144835089523,$
$0.449351036720358, 0.488537764956717, 0.443399636350081,$
$0.464692247253674, 0.485595145224172)$

g03

$$\min f(x) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$$

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

$n = 10, 0 \leq x_i \leq 1, i = 1, \cdots, n$

Simulation results obtained using CVI-PSO:

$best = -1.000000000000000$
$mean = -0.9999999999999999$
$worst = -0.9999999999999999$
$std.\ dev. = 0.37e-15$
$h_1 = 0$ (active)
$x^* = (0.316227764114971, 0.316227759153403, 0.316227770530290,$
$0.316227768137149, 0.316227761243207, 0.316227769973661,$
$0.316227764616772, 0.316227768908145, 0.316227766252533,$
$0.316227767238248)$

g04

$$\min f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.000626x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$
$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.000626x_1x_4 + 0.0022053x_3x_5 \leq 0$
$g_3(x) = 80.5124 + 0.007131x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$
$g_4(x) = -80.5124 - 0.007131x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$
$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$
$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$

$78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45, 27 \leq x_i \leq 45, i = 3, 4, 5$

Simulation results obtained using CVI-PSO:

$best = -30665.8217102204$
$mean = -30665.8209961102$
$worst = -30665.8032411366$
$std.\ dev. = 0.00339162403921788$
$g_1 = 0$ (active)
$g_2 = -92$
$g_3 = -11.15959996291961$
$g_4 = -8.840400037080386$
$g_5 = -5$

$g_6 = 0$ (active)
$x^* = (78, 33, 29.99344827103850, 45, 36.78038449548394)$

g05

$\min f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$
$g_1(x) = -x_4 + x_3 - 0.55 \leq 0$
$g_2(x) = -x_3 + x_4 - 0.55 \leq 0$
$h_1(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$
$h_2(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$
$h_3(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$
$0 \leq x_1 \leq 1200, 0 \leq x_2 \leq 1200, -0.55 \leq x_3 \leq 0.55, -0.55 \leq x_4 \leq 0.55$
Simulation results obtained using CVI-PSO:
$best = 5127.27766734565$
$mean = 5127.27766734565$
$worst = 5127.27766734565$
$std.\ dev. = 0$
$g_1 = -0.039438919371359$
$g_2 = -1.060561080628641$
$h_1 = 0$ (active)
$h_2 = 0$ (active)
$h_3 = 0$ (active)
$x^* = (692.1292821398356, 1013.079960815408, 0.1102023179908734,$
$-0.4003587626377678)$

g06

$\min f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$
$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$
$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$
$13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$
Simulation results obtained using CVI-PSO:
$best = -6961.813875580167$
$mean = -6961.813875580167$
$worst = -6961.813875580167$
$std.\ dev. = 0$
$g_1 = 0$ (active)
$g_2 = 0$ (active)
$x^* = (14.09499999999999, 0.8429607892154538)$

g07

$\min f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 +$
$2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$
$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$
$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$
$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$
$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$
$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$

$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$
$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$
$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$
$-10 \leq x_i \leq 10, i = 1, \cdots, 10$

Simulation results obtained using CVI-PSO:

$best = 24.4738268493991$
$mean = 26.5612953717409$
$worst = 29.5242543022246$
$std.\ dev. = 1.64251888141461$
$g_1 = -2.799777217887822e-05$ (active)
$g_2 = -1.338452356769437e-04$ (active)
$g_3 = -7.795774925867249e-05$ (active)
$g_4 = -4.729901273352311$
$g_5 = -1.700333459808689e-04$ (active)
$g_6 = -0.008209630144058$
$g_7 = -6.555562504262870$
$g_8 = -49.49952913797831$
$x^* = (2.200454319500069, 2.375451024469502, 8.635767668973507,$
$5.080522747470294, 0.941294600303058, 1.309352875839038,$
$1.330962032680838, 9.812642855546951, 8.296631014528680,$
$8.315250261665556)$

g08

$$\min f(x) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

$g_1(x) = x_1^2 - x_2 + 1 \leq 0$
$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$
$0 \leq x_1 \leq 10, 0 \leq x_2 \leq 10$

Simulation results obtained using CVI-PSO:

$best = -0.105459505088416$
$mean = -0.105459505088416$
$worst = -0.105459505088416$
$std.\ dev. = 0$
$g_1 = -1.237374596358008$
$g_2 = -0.162744489877962$
$x^* = (1.227816474163547, 3.744907890585411)$

g09

$\min f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 +$
$x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$
$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$
$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$
$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$
$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$
$-10 \leq x_i \leq 10, i = 1, \cdots, 7$

Simulation results obtained using CVI-PSO:

$best = 680.635400792284$

$mean = 680.755705150086$
$worst = 680.863957829483$
$std.\ dev. = 0.07923261328812$
$g_1 = -1.643130076445232e-14$ (active)
$g_2 = -252.3520244717333$
$g_3 = -144.7793230649124$
$g_4 = -1.350031197944190e-13$ (active)
$x^* = (2.334604927558698, 1.949963826853384, -0.4972308118012186, 4.367556810712299, -0.6159079420383568, 1.047061360328383, 1.606952534819435)$

g10

$\min f(x) = x_1 + x_2 + x_3$
$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$
$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$
$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$
$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$
$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$
$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$
$100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000\ (i = 2, 3)$,
$10 \leq x_i \leq 1000\ (i = 4, \cdots 8)$
Simulation results obtained using CVI-PSO:
$best = 7049.276585517206$
$mean = 7053.214310599261$
$worst = 7091.880859119432$
$std.\ dev. = 10.6156427859157$
$g_1 = 0$ (active)
$g_2 = 0$ (active)
$g_3 = 0$ (active)
$g_4 = -1.455191522836685e-11$ (active)
$g_5 = -2.910383045673370e-11$ (active)
$g_6 = -1.164153218269348e-10$ (active)
$x^* = (585.3598210908839, 1360.985804012512, 5102.930960413811, 182.5210948331164, 295.8827615834476, 217.4789051668836, 286.6383332496689, 395.8827615834476)$

g11

$\min f(x) = x_1^2 + (x_2 - 1)^2$
$h_1(x) = x_2 - x_1^2 = 0$
$-1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1$
Simulation results obtained using CVI-PSO:
$best = 0.750000000000000$
$mean = 0.750000000000000$
$worst = 0.750000000000000$
$std.\ dev. = 0$
$h_1 = 0$ (active)
$x^* = (-0.7071067812000658, 0.5000000000191177)$

$x^* = (0.7071067812740076, 0.5000000001236873)$

g12

$\min f(x) = -\frac{1}{100}\Big(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2\Big)$

$g_1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

$0 \leq x_i \leq 10, i = 1, 2, 3,$ and $p, q, r = 1, 2, \cdots, 9$

Simulation results obtained using CVI-PSO:

$best = -1.000000000000000$

$worst = -1.000000000000000$

$mean = -1.000000000000000$

$std.\ dev. = 0$

$g_1 = -0.062499999999993$

$x^* = (5.000000045858417, 5.000000055479061, 4.999999970903518)$

corresponding to : $p = 5,\ q = 5,\ r = 5$

g13

$\min f(x) = e^{x_1 x_2 x_3 x_4 x_5}$

$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$

$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0$

$h_3(x) = x_1^3 + x_2^3 + 1 = 0$

$-2.3 \leq x_1, x_2 \leq 2.3, -3.2 \leq x_3, x_4, x_5 \leq 3.2$

Simulation results obtained using CVI-PSO:

$best = 5.55558297420899e - 02$

$mean = 6.55907443993130e - 02$

$worst = 9.37281570016601e - 02$

$std.\ dev. = 1.01694200123061e - 02$

$h_1 = 0$ (active)

$h_2 = 0$ (active)

$h_3 = 1.332267629550188e - 15$ (active)

$x^* = (-1.666553133635026, 1.536678565048140, 1.916323688993618,$
$-0.821698873113951, -0.716752482673076)$

g14

$\min f(x) = \sum_{i=1}^{10} \left(x_i \Big(c_i - \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \Big) \right)$

$h_1(x) = x_1 + 2x_2 + +2x_3 + x_6 + x_{10} - 2 = 0$

$h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$

$h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$

$0 \leq x_i \leq 10, i = 1, \cdots, 10$

$c = (-6.089, -17.164, -34.054, -5.914, -24.721, -14.986, -24.1,$
$-10.708, -26.662, -22.179)$

Simulation results obtained using CVI-PSO:

$best = -47.453011430881$

$mean = -44.424690437261$

$worst = -42.427727115968$

$std.\ dev. = 1.406368088853$
$h_1 = 0$ (active)
$h_2 = 0$ (active)
$h_3 = 0$ (active)
$x^* = (0.0001, 0.019661041433836, 0.980188958566163, 0.0001, 0.4998500,$
$0.0001, 0.0001, 0.019411041433836, 0.0001, 0.0001)$

g15

$\min f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$
$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$
$h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$
$0 \leq x_i \leq 3, i = 1, \cdots, 10$
Simulation results obtained using CVI-PSO:
$best = 961.7157071471469$
$mean = 961.7185954748234$
$worst = 961.7187814063919$
$std.\ dev. = 6.879792947654714e - 04$
$h_1 = 0$ (active)
$h_2 = 0$ (active)
$x^* = (3.530984163806204, 0.215539725213053, 3.533510076652518)$

g16

$$\min f(x) = -0.0000005843y_{17} + 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.0000011502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}}$$

$g_1(x) = y_4 - \frac{0.28}{0.72}y_5 \geq 0$
$g_2(x) = 1.5x_2 - x_3 \geq 0$
$g_3(x) = 21 - 3496\frac{y_2}{c_{12}} \geq 0$
$g_4(x) = \frac{62.212}{c_{17}} - 110.6 - y_1 \geq 0$
$g_5(x), g_6(x) = 213.1 \leq y_1 \leq 405.23$
$g_7(x), g_8(x) = 17.505 \leq y_2 \leq 1053.6667$
$g_9(x), g_{10}(x) = 11.275 \leq y_3 \leq 35.03$
$g_{11}(x), g_{12}(x) = 214.228 \leq y_4 \leq 665.585$
$g_{13}(x), g_{14}(x) = 7.458 \leq y_5 \leq 584.463$
$g_{15}(x), g_{16}(x) = 0.961 \leq y_6 \leq 265.916$
$g_{17}(x), g_{18}(x) = 1.612 \leq y_7 \leq 7.046$
$g_{19}(x), g_{20}(x) = 0.146 \leq y_8 \leq 0.222$
$g_{21}(x), g_{22}(x) = 107.99 \leq y_9 \leq 273.366$
$g_{23}(x), g_{24}(x) = 922.693 \leq y_{10} \leq 1286.105$
$g_{25}(x), g_{26}(x) = 926.832 \leq y_{11} \leq 1444.046$
$g_{27}(x), g_{28}(x) = 18.766 \leq y_{12} \leq 537.141$
$g_{29}(x), g_{30}(x) = 1072.163 \leq y_{13} \leq 3247.039$
$g_{31}(x), g_{32}(x) = 8961.448 \leq y_{14} \leq 26844.086$
$g_{33}(x), g_{34}(x) = 0.063 \leq y_{15} \leq 0.386$
$g_{35}(x), g_{36}(x) = 71084.33 \leq y_{16} \leq 140000$
$g_{37}(x), g_{38}(x) = 2802713 \leq y_{17} \leq 12146108$
where:

$$
\begin{aligned}
y_1 &= x_2 + x_3 + 41.6 \\
c_1 &= 0.024x_4 - 4.62 \\
y_2 &= \frac{12.5}{c_1} + 12 \\
c_2 &= 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
c_3 &= 0.052x_1 + 78 + 0.002377y_2x_1 \\
y_3 &= \frac{c_2}{c_3} \\
y_4 &= 19y_3 \\
c_4 &= 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} \\
c_5 &= 100x_2 \\
c_6 &= x_1 - y_3 - y_4 \\
c_7 &= 0.950 - \frac{c_4}{c_5} \\
y_5 &= c_6c_7 \\
y_6 &= x_1 - y_5 - y_4 - y_3 \\
c_8 &= (y_5 + y_4)0.995 \\
y_7 &= \frac{c_8}{y_1} \\
y_8 &= c_83798 \\
c_9 &= y_7 - \frac{0.0663y_7}{y_8} - 0.3153 \\
y_9 &= \frac{96.82}{c_9} + 0.321y_1 \\
y_{10} &= 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6 \\
y_{11} &= 1.71x_1 - 0.452y_4 + 0.580y_3 \\
c_{10} &= \frac{12.3}{752.3} \\
c_{11} &= (1.75y_2)(0.995x_1) \\
c_{12} &= 0.995y_{10} + 1998 \\
y_{12} &= c_{10}x_1 + \frac{c_{11}}{c_{12}} \\
y_{13} &= c_{12} - 1.75y_2 \\
y_{14} &= 3623 + 64.4x_2 + 58.4x_3 + \frac{146.312}{y_9 + x_5} \\
c_{13} &= 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095 \\
y_{15} &= \frac{y_{13}}{c_{13}} \\
y_{16} &= 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13} \\
c_{14} &= 2324y_{10} - 28740000y_2 \\
y_{17} &= 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}} \\
c_{15} &= \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52} \\
c_{16} &= 1.104 - 0.72y_{15} \\
c_{17} &= y_9 + x_5 \\
&704.4148 \leq x_1 \leq 906.3855, 68.6 \leq x_2 \leq 288.88, 0 \leq x_3 \leq 134.75, \\
&193 \leq x_4 \leq 287.0966, 25 \leq x_5 \leq 84.1988
\end{aligned}
$$

g17

$$
\min f(x) = f_1(x_1) + f_2(x_2)
$$

$$
f_1(x_1) = \begin{cases} 30x_1 \text{ if } 0 \leq x_1 < 300 \\ 31x_1 \text{ if } 300 \leq x1 < 400 \end{cases}
$$

$$
f_2(x_2) = \begin{cases} 28x_2 \text{ if } 0 \leq x_2 < 100 \\ 29x_2 \text{ if } 100 \leq x_2 < 200 \\ 30x_2 \text{ if } 200 \leq x_2 < 1000 \end{cases}
$$

$$
h_1(x) = -x_1 + 300 - \frac{x_3x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588) = 0
$$

$$
h_2(x) = -x_2 - \frac{x_3x_4}{131.078}\cos(1.48477 + x6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588) = 0
$$

$h_3(x) = -x_5 - \frac{x_3 x_4}{131.078}\sin(1.48477 + x6) + \frac{0.90798 x_4^2}{131.078}\sin(1.47588) = 0$
$h_4(x) = 200 - \frac{x_3 x_4}{131.078}\sin(1.48477 - x6) + \frac{0.90798 x_3^2}{131.078}\sin(1.47588) = 0$
$0 \le x_1 \le 400, 0 \le x_2 \le 1000, 340 \le x_3 \le 420, 340 \le x_4 \le 420,$
$-1000 \le x_5 \le 1000, 0 \le x_6 \le 0.5236$

Simulation results obtained using CVI-PSO:

$best = 8853.539891329589$
$mean = 8853.539891329592$
$worst = 8853.539891329594$
$std.\ dev. = 3.700170984052295e - 12$
$h_1 = 0$ (active)
$h_2 = 0$ (active)
$h_3 = 0$ (active)
$h_4 = 0$ (active)
$x^* = (201.7846630443196, 99.99999999999999, 383.0709952729385,$
$419.9999999999998, -10.9076056293656, 0.073148148400340)$

g18

$\min f(x) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7)$
$g_1(x) = 1 - x_3^2 - x_4^2 \le 0$
$g_2(x) = 1 - x_9^2 \le 0$
$g_3(x) = 1 - x_5^2 - x_6^2 \le 0$
$g_4(x) = 1 - x_1^2 - (x_2 - x_9)^2 \le 0$
$g_5(x) = 1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \le 0$
$g_6(x) = 1 - (x_1 - x_7)^2 - (x_2 - x_8)^2 \le 0$
$g_7(x) = 1 - (x_3 - x_5)^2 - (x_4 - x_6)^2 \le 0$
$g_8(x) = 1 - (x_3 - x_7)^2 - (x_4 - x_8)^2 \le 0$
$g_9(x) = 1 - x_7^2 - (x_8 - x_9)^2 \le 0$
$g_{10}(x) = x_1 x_4 - x_2 x_3 \le 0$
$g_{11}(x) = x_3 x_9 \le 0$
$g_{12}(x) = -x_5 x_9 \le 0$
$g_{13}(x) = x_5 x_8 - x_6 x_7 \le 0$
$-10 \le x_i \le 10, i = 1, \cdots, 8$ and $0 \le x_9 \le 20$

Simulation results obtained using CVI-PSO:

$best = -0.8646312878998893$
$mean = -0.80910925896692$
$worst = -0.644436847288925$
$std.\ dev. = 0.0627009266838719$
$g_1 = -4.637481473279692e - 07$ (active)
$g_2 = -1.000000000000000$
$g_3 = -1.218304956918104e - 08$ (active)
$g_4 = -0.009414661842899$
$g_5 = -0.999961246806876$
$g_6 = -1.015298956019706e - 13$ (active)
$g_7 = -2.675193400136777e - 12$ (active)
$g_8 = -0.999998298342329$
$g_9 = -1.582247632933020e - 04$ (active)
$g_{10} = -0.86395737717531$

$g_{11} = 0$ (active)
$g_{12} = 0$ (active)
$g_{13} = -0.86530519862447$
$x^* = (0.1182510710879312, -0.988231765500208, 0.9208911742582345,$
$-0.389819165030585, 0.1228521787212526, -0.992424974494491,$
$0.9203101912114871, -0.390987118955322, 0)$

g19

$$\min f(x) = -\sum_{i=1}^{10} b_i x_i + \sum_{j=1}^{5}\sum_{i=1}^{5} c_{ij} x_{(10+i)} x_{(10+j)} + 2\sum_{j=1}^{5} d_j x_{(10+j)}^3$$

$$g_j(x) = 2\sum_{i=1}^{5} c_{ij} x_{(10+i)} + 3d_j x_{(10+j)}^2 + e_j - \sum_{i=1}^{10} a_{ij} x_i \geq 0, \quad j = 1, \cdots, 5$$

$0 \leq x_i \leq 10, i = 1, \cdots, 15$
The problem data is on the following table.

j	1	2	3	4	5	j	1	2	3	4	5		
e_j	-15	-27	-36	-18	-12	a_{1j}	-16	2	0	1	0	b_1	-40
c_{1j}	30	-20	-10	32	-10	a_{2j}	0	-2	0	0.4	2	b_2	-2
c_{2j}	-20	39	-6	-31	32	a_{3j}	-3.5	0	2	0	0	b_3	-0.25
c_{3j}	-10	-6	10	-6	-10	a_{4j}	0	-2	0	-4	-1	b_4	-4
c_{4j}	32	-31	-6	39	-20	a_{5j}	0	-9	-2	1	-2.8	b_5	-4
c_{5j}	-10	32	-10	-20	30	a_{6j}	2	0	-4	0	0	b_6	-1
						a_{7j}	-1	-1	-1	-1	-1	b_7	-40
d_j	4	8	10	6	2	a_{8j}	-1	-2	-3	-2	-1	b_8	-60
						a_{9j}	1	2	3	4	5	b_9	5
						a_{10j}	1	1	1	1	1	b_{10}	1

Simulation results obtained using CVI-PSO:
$best = 32.82702708996916$
$mean = 35.0673371976379$
$worst = 43.3749595963395$
$std.\ dev. = 2.2867287734165$
$g_1 = -3.890221478286549e - 12$ (active)
$g_2 = -1.400835003551038e - 11$ (active)
$g_3 = -1.256995219023338e - 07$ (active)
$g_4 = -6.389789142247082e - 09$ (active)
$g_5 = -0.417872705020541$
$x^* = (0, 0, 3.803210550534287, 0, 3.086077390641635, 10, 0, 0, 0, 0,$
$0.4050918734367448, 0.327746955799119, 0.545910893084831,$
$0.392524542408481, 0.283970853232303)$

g20

$$\min f(x) = \sum_{i=1}^{24} a_i x_i$$

$$g_i(x) = \frac{x_i + x_{(i+12)}}{\sum_{j=1}^{24} x_j + e_i} \leq 0, \quad i = 1, 2, 3$$

$$g_i(x) = \frac{x_{(i+3)} + x_{(i+15)}}{\sum_{j=1}^{24} x_j + e_i} \leq 0, \quad i = 4, 5, 6$$

$$h_i(x) = \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0, \quad i = 1, \cdots, 12$$

$$h_{13}(x) = \sum_{i=1}^{24} x_i - 1 = 0$$

$$h_{14}(x) = \sum_{i=1}^{12} \frac{x_i}{d_i} + (0.7302)(530)\Big(\frac{14.7}{40}\Big) \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0$$

$0 \leq x_i \leq 10, i = 1, \cdots, 24$

The problem data is on the following table.

i	a_i	b_i	c_i	d_i	e_i
1	0.0693	44.094	123.7	31.244	0.1
2	0.0577	58.12	31.7	36.12	0.3
3	0.05	58.12	45.7	34.784	0.4
4	0.2	137.4	14.7	92.7	0.3
5	0.26	120.9	84.7	82.7	0.6
6	0.55	170.9	27.7	91.6	0.3
7	0.06	62.501	49.7	56.708	
8	0.1	84.94	7.1	82.7	
9	0.12	133.425	2.1	80.8	
10	0.18	82.507	17.7	64.517	
11	0.1	46.07	0.85	49.4	
12	0.9	60.097	0.64	49.1	
13	0.0693	44.094			
14	0.0577	58.12			
15	0.05	58.12			
16	0.2	137.4			
17	0.26	120.9			
18	0.55	170.9			
19	0.06	62.501			
20	0.1	84.94			
21	0.12	133.425			
22	0.18	82.507			
23	0.1	46.07			
24	0.09	60.097			

Simulation results obtained using CVI-PSO:

$best = 0.2146896719462919$

$mean = NA$

$worst = NA$

$std.\ dev. = NA$

$g_1 = 0$ (active)

$g_2 = 0$ (active)

$g_3 = 0$ (active)

$g_4 = 0$ (active)
$g_5 = 0$ (active)
$g_6 = 0$ (active)
$h_1 = 0$ (active)
$h_2 = 0$ (active)
$h_3 = 0$ (active)
$h_4 = 0$ (active)
$h_5 = 1.832400897683328e-11$ (active)
$h_6 = 0$ (active)
$h_7 = 0$ (active)
$h_8 = 0$ (active)
$h_9 = 0$ (active)
$h_{10} = 7.626443920827342e-11$ (active)
$h_{11} = 0$ (active)
$h_{12} = 0$ (active)
$h_{13} = -4.990663438064757e-11$ (active)
$h_{14} = 8.607521362336001e-10$ (active)
$x^* = (0, 0, 0, 0, 0.4088644406169757, 0, 0, 0, 0, 0.5593021601635705, 0, 0,$
$0, 0, 0, 0, 0.0247564588239630, 0, 0, 0, 0, 0.0070769403455842, 0, 0)$

g21

$\min f(x) = x_1$
$g_1(x) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \leq 0$
$h_1(x) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0$
$h_2(x) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0$
$h_3(x) = -x_5 + \ln(-x_4 + 900) = 0$
$h_4(x) = -x_6 + \ln(x_4 + 300) = 0$
$h_5(x) = -x_7 + \ln(-2x_4 + 700) = 0$
$0 \leq x_1 \leq 1000, 0 \leq x_2, x_3 \leq 40, 100 \leq x_4 \leq 300, 6.3 \leq x_5 \leq 6.7,$
$5.9 \leq x_6 \leq 6.4, 4.5 \leq x_7 \leq 6.25$
Simulation results obtained using CVI-PSO:
$best = 193.7869252468235$
$mean = 193.7869352259308$
$worst = 193.7870933708868$
$std.\ dev. = 3.384701509121538e-05$
$g_1 = -1.091962076316122e-09$ (active)
$h_1 = 4.547473508864641e-13$ (active)
$h_2 = 0$ (active)
$h_3 = 0$ (active)
$h_4 = 0$ (active)
$h_5 = 0$ (active)
$x^* = (193.7869252468235, 0, 17.32848970613118, 100.0020246198148,$
$6.684609196889957, 5.991469608644709, 6.214599999910139)$

g22

$\min f(x) = x1$
$g_1(x) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$

$h_1(x) = x_5 - 100000x_8 + 1e{+}7 = 0$
$h_2(x) = x_6 + 100000x_8 - 100000x_9 = 0$
$h_3(x) = x_7 + 100000x_9 - 5e{+}7 = 0$
$h_4(x) = x_5 + 100000x_{10} - 3.3e{+}7 = 0$
$h_5(x) = x_6 + 100000x_{11} - 4.4e{+}7 = 0$
$h_6(x) = x_7 + 100000x_{12} - 6.6e{+}7 = 0$
$h_7(x) = x_5 - 120x_2x_{13} = 0$
$h_8(x) = x_6 - 80x_3x_{14} = 0$
$h_9(x) = x_7 - 40x_4x_{15} = 0$
$h_{10}(x) = x_8 - x_{11} + x_{16} = 0$
$h_{11}(x) = x_9 - x_{12} + x_{17} = 0$
$h_{12}(x) = -x_{18} + \ln(x_{10} - 100) = 0$
$h_{13}(x) = -x_{19} + \ln(-x_8 + 300) = 0$
$h_{14}(x) = -x_{20} + \ln(x_{16}) = 0$
$h_{15}(x) = -x_{21} + \ln(-x_9 + 400) = 0$
$h_{16}(x) = -x_{22} + \ln(x_{17}) = 0$
$h_{17}(x) = -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0$
$h_{18}(x) = x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0$
$h_{19}(x) = x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0$
$0 \leq x_1 \leq 20000, 0 \leq x_2, x_3, x_4 \leq 1e{+}6, 0 \leq x_5, x_6, x_7 \leq 4e{+}7,$
$100 \leq x_8 \leq 299.99, 100 \leq x_9 \leq 399.99, 100.01 \leq x_{10} \leq 300,$
$100 \leq x_{11} \leq 400, 100 \leq x_{12} \leq 600, 0 \leq x_{13}, x_{14}, x_{15} \leq 500,$
$0.01 \leq x_{16} \leq 300, 0.01 \leq x_{17} \leq 400, -4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$

Simulation results obtained using CVI-PSO:

$best = NA$
$mean = NA$
$worst = NA$
$std.\ dev. = NA$

g23

$\min f(x) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$
$g_1(x) = x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0$
$g_2(x) = x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0$
$h_1(x) = x_1 + x_2 - x_3 - x_4 = 0$
$h_2(x) = 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0$
$h_3(x) = x_3 + x_6 - x_5 = 0$
$h_4(x) = x_4 + x_7 - x_8 = 0$
$0 \leq x_1, x_2, x_6 \leq 300, 0 \leq x_3, x_5, x_7 \leq 100,$
$0 \leq x_4, x_8 \leq 200, 0.01 \leq x_9 \leq 0.03$

Simulation results obtained using CVI-PSO:

$best = -400$
$worst = -400$
$mean = -400$
$std.\ dev. = 0$
$g_1 = 0$ (active)
$g_2 = 0$ (active)
$h_1 = 0$ (active)
$h_2 = 0$ (active)

$h_3 = 0$ (active)

$h_4 = 0$ (active)

$x^* = (0, 100, 0, 100, 0, 0, 100, 200, 0.01)$

g24

$\min f(x) = -x_1 - x_2$

$g_1(x) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$

$g_2(x) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$

$0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4$

Simulation results obtained using CVI-PSO:

$best = -5.508013271595327$

$worst = -5.508013271595297$

$mean = -5.508013271595326$

$std.\ dev. = 9.465924043274113e - 15$

$g_1 = -1.776356839400251e - 14$ (active)

$g_2 = 0$ (active)

$x^* = (2.329520197477613, 3.178493074117714)$

Pressure vessel design problem

$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

$g_1(x) = -x_1 + 0.0193x_3 \leq 0$

$g_2(x) = -x_2 + 0.00954x_3 \leq 0$

$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0$

$g_4(x) = x_4 - 240 \leq 0$

$0 \leq x_1, x_2 \leq 100, 10 \leq x_3, x_4 \leq 200$

Simulation results obtained using CVI-PSO:

$best = 6059.7143$

$mean = 6292.1231$

$worst = 6820.4101$

$std.\ dev. = 288.4550$

$g_1 = -0.0000000$ (active)

$g_2 = -0.0358808$

$g_3 = -0.0000000$ (active)

$g_4 = -63.363404$

$x^* = (0.8125, 0.4375, 42.0984455958, 176.636595842)$

Table 3 Comparison of the best solution for pressure vessel problem by different algorithms

	CVI-PSO	T-Cell [4]	AATM [2]	HPSO [3]	CPSO [1]
x_1	0.8125	0.8125	0.8125	0.8125	0.812500
x_2	0.4375	0.4375	0.4375	0.4375	0.437500
x_3	42.0984455958	42.098429	42.0983827	42.0984	42.091266
x_4	176.636595842	190.78769	176.637528	176.6366	176.7465
g_1	**−1.059929921609637e−12**	**−0.00000**	**−1.2e−06**	**−8.8e−07**	−0.000139
g_2	−0.0358808290160679	−0.035881	−0.035881	−0.0359	−0.035949
g_3	**6.235670298337936e−06**	−0.114149	−0.857920	−3.1227	−116.3827
g_4	−63.36340415800001	−430.78769	−63.362471	−63.3634	−63.25350
$f(x)$	**6059.7143**	6390.554	6059.7255	**6059.7143**	6061.0777

Table 4 Statistical results of different algorithms for pressure vessel design problem

	FFE#	best	mean	worst	std. dev.
CPSO [1]	200000	6061.0777	6147.1332	6363.8041	86.4545
HPSO [3]	81000	**6059.7143**	6099.9323	6288.6770	86.2022
AATM [2]	**30000**	6059.7255	**6061.9878**	**6090.8022**	**4.7000**
T-Cell [4]	81000	6390.5540	6737.0651	7694.0668	357.0000
CVI-PSO	**25000**	**6059.7143**	6292.1231	6820.4101	288.4550

Welded beam design problem

$$\min f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

$$g_1(x) = \tau(x) - 13600 \leq 0$$
$$g_2(x) = \sigma(x) - 30000 \leq 0$$
$$g_3(x) = x_1 - x_4 \leq 0$$
$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$$
$$g_5(x) = 0.125 - x_1 \leq 0$$
$$g_6(x) = \delta(x) - 0.25 \leq 0$$
$$g_7(x) = P - P_c(x) \leq 0$$
$$0.1 \leq x_1, x_4 \leq 2, 0.1 \leq x_2, x_3 \leq 10$$

where:

$$P = 6000,\ L = 14,\ E = 30 \cdot 10^6,\ G = 12 \cdot 10^6$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2},\ \delta(x) = \frac{6PL^3}{Ex_3^3x_4},\ Q = P\left(L + \frac{x_2}{2}\right),\ R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{QR}{J},\ J = 2\left(\sqrt{2}x_1x_2\left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)\right)$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\ \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$$

Simulation results obtained using CVI-PSO:

$best = 1.724852$
$mean = 1.725124$
$worst = 1.727665$
$std.\ dev. = 6.1205e-04$
$g_1 = -3.7107e-10$ (active)
$g_2 = -1.1641e-10$ (active)
$g_3 = -4.9960e-15$ (active)
$g_4 = -3.432983$
$g_5 = -0.080729$
$g_6 = -0.228310$
$g_7 = -1.8189e-12$ (active)
$x^* = (0.2057296397, 3.4704886656, 9.0366239103, 0.2057296397)$

Table 5 Comparison of the best solution for welded beam problem by different algorithms

	CVI-PSO	HPSO [3]	CPSO [1]	CGA[7]	GAPF[6]
x_1	0.2057296397	0.205730	0.202369	0.208800	0.248900
x_2	3.4704886656	3.470489	3.544214	3.420500	6.173000
x_3	9.0366239103	9.036624	9.048210	8.997500	8.178900
x_4	0.2057296397	0.205730	0.205723	0.210000	0.253300
g_1	**5.950962076894939e − 06**	−0.025399	−12.83979	−0.337812	−5758.604
g_2	**1.293499008170329e − 05**	−0.053122	−1.247467	−353.9026	−255.5769
g_3	**0**	**−0.000000**	−0.001498	−0.001200	−0.004400
g_4	−3.432983786032229	−3.432981	−3.429347	−3.411865	−2.982866
g_5	−8.072963969999999e − 02	−0.080730	−0.079381	−0.083800	−0.123900
g_6	−0.228310483867641	−0.228310	−0.235536	−0.235649	−0.234160
g_7	**7.556549462606199e − 06**	−0.031555	−11.68135	−363.2324	−4465.271
$f(x)$	**1.724852**	**1.724852**	1.728024	1.748309	2.433116

Table 6 Statistical results of different algorithms for welded beam design problem

	FFE#	best	mean	worst	std. dev.
GAPF[6]	N/A	2.433116	N/A	N/A	N/A
CGA[7]	900000	1.748309	1.771973	1.785835	0.011220
CPSO [1]	200000	1.728024	1.748831	1.782143	0.012926
HPSO [3]	81000	**1.724852**	1.749040	1.814295	0.040049
CVI-PSO	**25000**	**1.724852**	**1.725124**	**1.727665**	**6.1205e − 04**

Tension/compression string design problem

$$\min f(x) = (x_3 + 2)x_2x_1^2$$

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0$$

$g_3(x) = 1 - \frac{140.45x_1}{x_2^2 x_3} \leq 0$
$g_4(x) = \frac{x_1+x_2}{1.5} - 1 \leq 0$
$0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 15$

Simulation results obtained using CVI-PSO:

$best = 0.0126652$
$mean = 0.0127310$
$worst = 0.0128426$
$std.\ dev. = 5.58e-05$
$g_1 = -1.553e-11$ (active)
$g_2 = -7.177e-12$ (active)
$g_3 = -5.823310300$
$g_4 = -1.091578331$
$x^* = (0.0516896544, 0.3567320142, 11.2881289355)$

Table 7 Comparison of the best solution for string design problem by different algorithms

	CVI-PSO	T-Cell [4]	AATM [2]	HPSO [3]	CPSO [1]
x_1	0.0516896544	0.051622	0.0518130955	0.051706	0.051728
x_2	0.3567320142	0.355105	0.3596904119	0.357126	0.357644
x_3	11.2881289355	11.384534	11.119252680	11.265083	11.244543
g_1	**$-1.731772725221958e-09$**	$-2.9e-05$	$-8.37774e-05$	$-1.574e-06$	-0.000845
g_2	**$1.308880115402644e-09$**	$-4.0e-06$	$-4.16824e-05$	$1.3916e-06$	$-1.26e-05$
g_3	-4.053813819756309	-4.050423	-5.838571569	-5.8254	-4.051300
g_4	-0.727718887600000	-0.728849	-1.088496492	-1.0912	-0.727090
$f(x)$	**0.01266523328**	**0.012665**	0.0126682620	**0.0126652**	0.0126747

Table 8 Statistical results of different algorithms for tension/compression string problem

	FFE#	best	mean	worst	std. dev.
CPSO [1]	200000	0.0126747	0.0127300	0.0129240	$5.1985e-05$
HPSO [3]	81000	**0.0126652**	**0.0127072**	**0.0127191**	**$1.5824e-05$**
AATM [2]	**25000**	0.0126682	0.0127080	0.0128613	$4.5e-05$
T-Cell [4]	**36000**	**0.012665**	0.012732	0.013309	$9.4e-05$
CVI-PSO	**25000**	**0.0126652**	0.0127310	0.0128426	$5.58e-05$

References

1. He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence* 2007; **20**(1):8999.
2. Wang Y, Cai Z, Zhou Y. Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. *International Journal for Numerical Methods in Engineering* 2009; **77**(11):15011534.
3. He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied mathematics and computation* 2007; **186**(2):14071422.

4. Aragon VS, Esquivel SC, Coello CAC. A modified version of a T-Cell Algorithm for constrained optimization problems. *International Journal for Numerical Methods in Engineering* 2010; **84**(3): 351378.
5. Mezura ME, Miranda-Varela ME, Gomez-Ramon RC. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences* 2010, **180**: 4223-4262.
6. Deb K. Optimal design of a welded beam via genetic algorithms. *AIAA Journal* 1991; **29**(11):20132015.
7. Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry* 2000; **41**:113127.